Tampere University

Omer Shafiq

ANOMALY DETECTION IN BLOCKCHAIN

Faculty of Information Technology and Communication Sciences
Masters Thesis
December 2019

Abstract

Omer Shafiq: Anomaly detection in blockchain
Master's thesis
Tampere University
Master's Degree Programme in Computational Big Data Analytics
December 2019

Anomaly detection has been a well-studied area for a long time. Its applications in the financial sector have aided in identifying suspicious activities of hackers. However, with the advancements in the financial domain such as blockchain and artificial intelligence, it is more challenging to deceive financial systems. Despite these technological advancements many fraudulent cases have still emerged.

Many artificial intelligence techniques have been proposed to deal with the anomaly detection problem; some results appear to be considerably assuring, but there is no explicit superior solution. This thesis leaps to bridge the gap between artificial intelligence and blockchain by pursuing various anomaly detection techniques on transactional network data of a public financial blockchain named 'Bitcoin'.

This thesis also presents an overview of the blockchain technology and its application in the financial sector in light of anomaly detection. Furthermore, it extracts the transactional data of bitcoin blockchain and analyses for malicious transactions using unsupervised machine learning techniques. A range of algorithms such as isolation forest, histogram based outlier detection (HBOS), cluster based local outlier factor (CBLOF), principal component analysis (PCA), K-means, deep autoencoder networks and ensemble method are evaluated and compared.

Keywords: blockchain, bitcoin, anomaly detection, unsupervised learning, fraud detection, deep learning

Contents

1 Introduction

Suspicious activities in network structures are as old as the invention of the network structure itself. Entities or their activities which tends to behave abnormally within the system is referred to as anomalies. Anomaly detection is widely used in a variety of cybersecurity applications for targeting financial fraud identification, network invasion detection, anti money laundering, virus detection, and more. Usually, a common goal in these networks is to detect those anomalies and prevent such illegal activities from happening in future. With advancements in technology, blockchain emerges to plays a vital role in securing these network structures. Carlozo (2017) elaborates that a blockchain is a decentralised distributed database that maintains on-growing records and ensures that fraudulent records do not become part of this database and previously added records stay immutable. However, participants of a blockchain network can try to conduct illegal activities and in some cases succeed in deceiving the system into their advantage.

Current state-of-the-art anomaly detection methodologies are designed and implemented in light of the centralised systems. With the advent of blockchain technology, it brings a need for anomaly detection procedures within these systems as well. In this thesis, we extract and analyse the data from a publicly available blockchain named bitcoin. Nakamoto (2008) states that bitcoin is a peer-to-peer digital currency blockchain, by using which users can send and receive a form of electronic cash to each other anonymously without the need for any intermediaries.

This thesis aims to detect anomalous or suspicious transactions in the bitcoin network, where all nodes are unlabeled, and there is no evidence that if any given transaction is an illicit activity. The primary focus is to detect anomalies within the bitcoin transaction network. Farren, Pham, and Alban-Hidalgo (2016) explained that the problem is related to the study of fraud detection in all types of financial transaction blockchain systems. The problem can also be generalised to other blockchain networks such as health service blockchains, public sector blockchains and many others, and therefore this thesis investigates a more general problem of anomaly detection in blockchains. Chapter 2 gives an in-depth background of blockchains and its procedures, and it also explains the problem statement along with some examples of problem use-cases. Chapter 3 discusses the history and development of anomaly detection research in the focus of distributed ledger and blockchains. Some research related to anomaly detection in financial systems is also discussed. Chap-

ter 4 explains in detail the unsupervised machine learning techniques that have been used for fraud detection in this thesis. Chapter 5 describes the technologies used to perform experiments and explains the data exploration and pre-processing along with the evaluation metrics used in this research. Chapter 6 describes the anomaly detection experiments and their evaluation results. The code for experiments is shared on Github[1]. The anomalies dataset[2], bitcoin transactional dataset[3] and bitcoin transaction network metadata[4] used in experiments are donated to IEEE Dataport. Chapter 7 details a comparison among all the evaluation results obtained from conducting experiments so we can narrow down to an optimal solution. Finally chapter 8 concludes the research by explaining what we found and what can be improved.

[1]https://github.com/epicprojects/blockchain-anomaly-detection
[2]https://ieee-dataport.org/open-access/bitcoin-hacked-transactions-2010-2013
[3]https://ieee-dataport.org/open-access/bitcoin-transactions-data-2011-2013
[4]https://ieee-dataport.org/open-access/bitcoin-transaction-network-metadata-2011-2013

2 Background

Use of ledger in the financial area is as ancient as money itself. From clay and stones to papyrus and paper and later to spreadsheets in the digital era, ledgers have been an essential part of accounting journey. Early digital ledgers have just merely mimicked the paper-based approaches of double entry book-keeping. However, still, there remained a vast gap in consensus when these ledgers grew in volume and spatiality. The rise in computing power and cryptography breakthroughs along with the discovery of some new and sophisticated algorithms have given birth to distributed ledgers.

2.1 Distributed ledger technology (DLT)

In the crudest form distribute ledger is merely a database whose copy is independently held by each participant who wants to update it. These participants usually are nodes on a network and records written on these databases are usually called transactions. Distribution of these records is unique as they are not communicated to other nodes in the network using a central authority node. However, instead of that, each record is independently composed by each node individually. This leads to all the participant nodes of the network to process all incoming transactions and reach a conclusion about its authenticity. Finally, a consensus is achieved based on majority votes of conclusions of the network. Once there is this consensus achieved the distributed ledger is updated, and at this point, all nodes on the network maintain an identical copy of the ledger which holds all transactions. This architecture enables a new dexterity as the system of records can go beyond being a simple database.

Distributed ledger is a dynamic type of media that possesses the attributes and capability go beyond legacy databases. The innovation of distributed ledgers allows its users not only to accumulate and communicate information in a distributedly secure manner but also enables them to go beyond relationships among data. Ibanez et al. (2017) state that distributed ledgers provide a trustworthy, secure, and accountable way of tracking transactions without the need for a central validating authority, and they could provide the foundation to make the web a genuinely decentralised autonomous system.

2.1.1 Blockchain

The blockchain is a type of distributed ledger. Apparent from its name it is a chain of logically connected data blocks. A list of growing records refer to as blocks, and each block contains a timestamp, transactional data and a unique cryptographic hash. The hash in the block links it to the previous block in the distributed ledger all the way to form a chain of logical links to the first block called genesis block using these cryptographic hashes (Figure 2.1). The blockchain is a discrete design which makes it resistant to modification of the data as well as duplicate entries. Iansiti and Lakhani (2017) said that it is an open-distributed ledger that can be utilised to document transactions among two unknown parties verifiably and permanently.

A blockchain usually uses a peer-to-peer network for seamless inter-node communication and validating new blocks. Once transaction data is appended to the block, it cannot be altered retroactively without modification of all subsequent blocks. In order to alter any record, it would require a consensus of the majority of network nodes, which is highly challenging to achieve. Although records of a blockchain are not immutable, still it may be considered the secure design and less prone to vulnerabilities. It also illustrates a distributed computing system with high byzantine fault tolerance. The decentralization and byzantine problem will be explained later in this chapter. Raval and Siraj (2016) state that the blockchain claims to have achieved decentralized consensus.

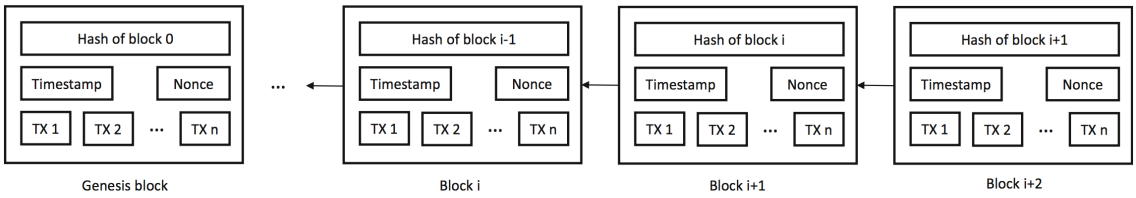

Figure 2.1. *Example of a blockchain; Z. Zheng et al. (2017)*

2.1.2 Hashgraph

In the world of distributed ledgers, blockchain has been taking all the spotlight. However, blockchain comes with some design limitations. A consensus mechanism called proof of work (POW) which will be discussed later in this chapter under subsection protocols of blockchain is widely in use. As Bitcoin-Wiki (2016) explains, proof of work algorithm can be considerably time-consuming in nature. An another inherited inefficiency of blockchain design is that it consumes considerable amount electricity even when it decides to discard the blocks.

As Baird (2016) explains, hashgraph algorithm is a consensus mechanism based on the directed acyclic graph (DAG) which uses a virtual voting system combined with a gossip protocol described on Wikipedia (2018) to achieve consensus in a faster, more reliable and secure manner. Hashgraph intends to provide the benefits of blockchain as a distributed ledger but without the limitations. Contrary to many ledgers which only use the gossip protocol, hashgraph utilises a voting algorithm in combination with gossip of gossips in order to reach consensus without using proof of work. This gossip of gossips is designed to share new information with other participant nodes of the network which are unaware of this new knowledge using hashgraph itself. Veronese et al. (2013) states that this guarantees Byzantine Fault Tolerance (BFT) as all participants are part of the gossip events. Burkhardt, Werling, and Lasi (2018) explain that hashgraph is DDoS attack resilient and provides high transaction throughput, low consensus latency and proper absolute ordering of transactions. It works asynchronously to nondeterministic achieve consensus using probabilities. Since hashgraph is an early bird patented solution and is not open-source, its commercial applications are still under consideration in industry.

2.2 Blockchain network

Blockchain networks define the communication channel of participant network nodes. Gossip peer-to-peer network is the most commonly used network, but with the rise in need of privacy, semi-gossiping networks and simple peer-to-peer networks are also utilised. A blockchain also defines its scope and protocols concerning its network with clarifies its intentions as a platform. Depending on the applications a blockchain scope and protocol may vary from a small scale node infrastructure to a massive hub platform. This thesis aims to propose a possible solution for all blockchain scopes, but due to data availability reasons, a public blockchain is under study.

2.2.1 Scopes of blockchain

As mentioned by the Lin, Liao, and Lin (2017) there are three significant scopes of a distributed ledger or a blockchain:

- **Public** - A state of the art public blockchain is open source and permissionless. Participants do not require any permission to join the network. Anyone can download its source code or binaries and run it locally on his or her computer and start validating transactions coming to the network, consequently joining the consensus process. This gives everyone on the network the ability to

process and determine which new blocks will become part of the chain. Participants of the network can send transactions onto the network and expect them to become part of the ledger as long as they are valid. In public blockchains transparency plays a key role, Each transaction on the blockchain ledger is visible to everyone and can be read using block explorer. However, these transactions are anonymous, so it is almost impossible to track the identity of the transaction owner. Milutinovi (2018) has explained Bitcoin, Ethereum and Litecoin which are some of the few examples of public blockchains.

• **Consortium** - Consortium blockchain regulates under the leadership of certain specific groups which follow same vision. Unlike public blockchains, they publicly do not allow everyone with internet access to become a participant of the network to verify the transactions. However, in some cases, the right to read the blockchain can be defined by the groups so that the access to the blockchain can be either public, restricted or hybrid. Consortium blockchains are highly scalable and fast and provide more transactional privacy, so their practical applications are more welcomed in industries. According to Buterin (2016), consortium blockchains are "partially decentralised".

Usage of consortium blockchains is often in association with enterprises. Collaborating group of companies leverage blockchain technology for improved business processes. Its application is already making waves in healthcare, supply chain, finance and other industries. Valenta and Sandner (2017) has mentioned that Quorum, R3 Corda and Hyperledger are some examples of consortium blockchains.

• **Private** - In most cases, private blockchains are developed, maintained and kept centralised under an organisation and are not open sourced. Read permissions can be either public, restricted or hybrid based on system requirements. In this closed environment, external public audits are required to maintain the integrity of the system. Private blockchains take advantage of the technology while still keeping the solution to themselves. A potential security risk is involved with the centralisation factor of private blockchains, but at the same time, it enables several advantages over public blockchains such as preapproved network participants and known identities. MONAX and Multichain are few known examples of private blockchains mentioned by Sajana, Sindhu, and Sethumadhavan (2018).

2.2.2 Consensus of blockchain

Skvorc (2018) elaborates that there are three primary consensus mechanisms of a distributed ledger or a blockchain:

• **Proof of work (POW)** - To keep the decentralised system secure and robust, proof of work plays an important role. The idea behind proof of work is to make participants of the network approve actual transactions and disapprove fraudulent ones. Approved transactions are added to a block which later becomes part of the blockchain ledger, and for this, the network rewards the participants. At core proof of work solely depends on computing power. Some participant nodes of the network engage in a competition known as the mining process to finding a hash called nonce (number used once). This hash is an extract of solving a complex mathematical problem that is part of the blockchain program. Combining this hash with the data in the block and then passing it through the hash function produces a result in a certain range which can become part of the blockchain. Since hash function makes it impossible to predict the output, so the participant nodes have to guess to find this hash. The node which finds the hash first is allowed to add its block to the blockchain and receive the reward. This reward usually comprises tokens that user can utilise on the same network. Amount of the reward is defined dynamically within the blockchain program and can mutate over time.

• **Proof of stake (POS)** - Proof of stake proposes to overcome the computing race in proof of work. In proof of work participants of the network can join together to create a pool of nodes in order to mine blocks faster and collect rewards. This network pool ends up utilising more electricity and approaches toward a more centralised network topology. Instead of letting all the participants competing with each other for mining blocks proof of stake uses an election process in which a participant is selected to validate the next block. These selected participants are called validators. To become a validator a node has to deposit a certain amount of tokens into the network as a stake of security collateral. Size of the stake determines the chances of a validator be chosen to forge the next block. A validator node checks for all the transactions within a block are valid and signs the block before adding it to the blockchain, consequently receiving the reward. Trust of the validator nodes depends on their deposited stake; validators will lose a part of their deposit if they approve fraudulent transactions.

• **Proof of Authority (POA)** - With popularity in private blockchains, proof of authority uses a voting algorithm to validate the blocks. A group of known and authorised nodes votes on which transaction should be approved and added to the block. This results in higher throughput and shorter verification time compared to proof of work. Many industries back this consensus mechanism due to its control over the system approvals. However, to reach the accurate decentralisation decisions of trusted participant nodes should not be influenced by anyone.

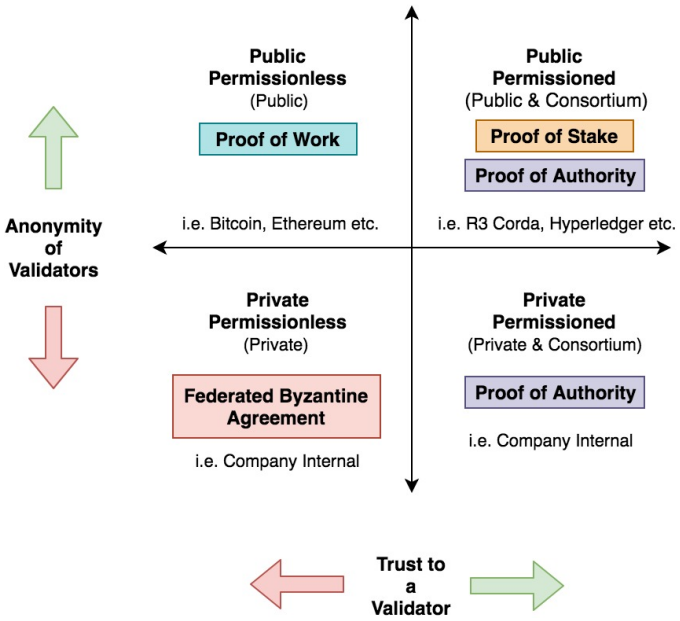

Figure 2.2. *Scopes and consensus mechanisms of a blockchain.*

2.3 Blockchain security

The core idea behind blockchain security is to allow people, particularly those who do not trust one another to share valuable information in a secure and tamper-proof manner. Sophisticated decentralised storage and consensus algorithms make it extremely difficult for attackers to manipulate the system. Encryption plays an essential role in the whole process and validates the communication channels of the network. Another critical problem blockchain claims to resolve is byzantine agreement. It defines the dependability of the system in case of failure.

2.3.1 Decentralization

Centrally held data is highly prone to a variety of risks. Blockchain eliminates this risk by storing data in a decentralised fashion on a peer-to-peer network. It is hard to exploit the vulnerabilities of a peer-to-peer network due to lack of centralised points. Moreover, it also ensures that there is no single point of failure in the system. Brito and Castillo (2013) explain that blockchain uses cryptographic public-private keys to communicate ensuring privacy and security of network nodes. Usage of private keys to encrypt and public keys to decrypt makes blockchain highly secure. The-Economist (2015) stated that generally, the data on blockchain is incorruptible. Data quality is also ensured with decentralisation as there is no official, centralised copy of a record and trust of each node is on the same level.

2.3.2 Byzantine Problem

A Byzantine problem, as explained by Lamport, Shostak, and Pease (1982), states that computers on a decentralised network cannot be entirely and irrefutably guaranteed that they are displaying the same data. Considering the unreliability of the network nodes can never be sure if the data they communicated has reached its destination. At its essence byzantine problem is about achieving consensus across a distributed network of nodes, while few nodes on the network could be possibly faulty and may also attempt to attack the network. A Byzantine node can mislead and may provide corrupted information to other nodes involved in the consensus process. Blockchain has to operate with robustness in such situation and reach consensus despite the interference of these byzantine nodes. Even though Byzantine problem cannot be fully solved, different blockchains and distributed ledgers utilise

various consensus mechanisms like proof of work and proof of stake to overcome the problem in a probabilistic manner.

2.4 Bitcoin

The earliest known design of a cryptographically chained data linked in chains and blocks was by Haber and Stornetta (1990). They purposed a system in which documents time stamps could not be tempered or backdated. Later on Bayer, Haber, and Stornetta (1993) added Merkle trees to the design which enhanced efficiency and allowed accumulation of multiple documents into a block. However, the first ever conceptualisation of a blockchain was introduced by a person with pseudonym Nakamoto (2008). Consequently, giving birth to the first blockchain system. This blockchain system was named 'Bitcoin' and is designed as peer-to-peer electronic cash system otherwise knows as a cryptocurrency. Bitcoin serves a public ledger for all transactions of the network and is capable of representing currency digitally hence working as electronic cash. A digital representation of an asset can lead to multiple problems. Bitcoin is developed to tackle some significant problems. Firstly, in theory, the digitally represented asset can technically be duplicated which is not acceptable. As this replication of a digital asset can confuse actual ownership of the asset. Secondly, the replicated asset can be spent multiple times resulting in chaos in the system, Usman W. Chohan (2017) states that this problem is also known as the double spend problem. Bitcoin solves this problem by using proof of work (POW) algorithm (Fig 2.3). Utilising proof of work mechanism miners verify transactions for their authenticity and get rewarded in newly generated bitcoins.

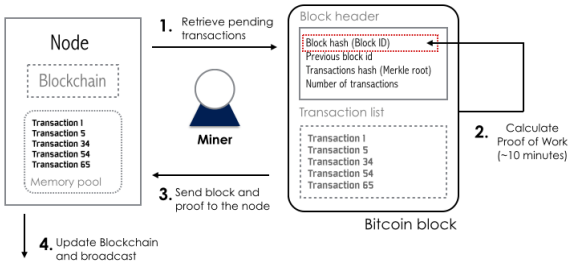

Figure 2.3. *Proof of work illustration; Barrera (2014)*

However, bitcoin blockchain is designed with some specific rules, such as there can be only 21 million bitcoins. As fig 2.4 illustrates, each transaction is encrypted with the private key of the sender. This transaction is enclosed with a digital signature and public key of the receiver and sent to the receiver address. This technique allows the asset to be securely delivered only to the receiver address and can be verified

using the digital signature. Identities of sender and receiver are anonymous as they can generate new addresses before initiating a new transaction.

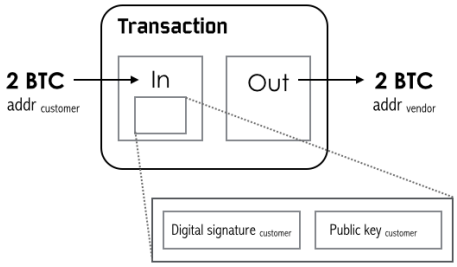

Figure 2.4. *Simple bitcoin transaction; Barrera (2014)*

Bitcoin transactions can consist of one or more inputs and one or more outputs. When sending bitcoins to an address, a user designates the address and the amount being sent and encapsulates it in the output of transaction in order to keep track of double spending. As mentioned by Delgado-Segura et al. (2018) it is known as unspent transaction output (UTXO) model, which implies that each transaction input is a pointer to the output of the previous transaction. Figure 2.5 shows an example where a customer sends 2 BTC to the address of the vendor, then the vendor sends 0.8 BTC of those 2 BTC to the provider address, and change of 1.2 BTC is the returned back to the vendor.

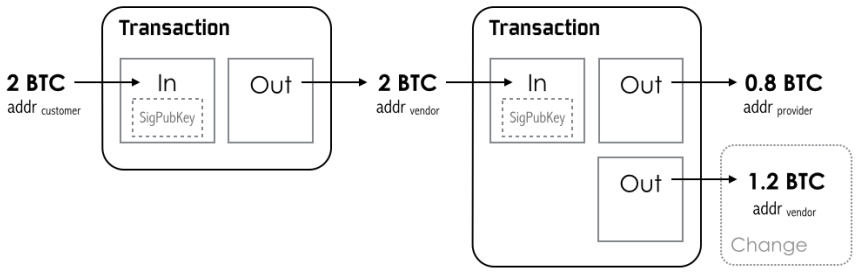

Figure 2.5. *Linked bitcoin transactions; Barrera (2014)*

There are many elements involved when participants of network transmit bitcoins. However, we can break down the activities in two groups; User activities and Transaction activities. In this thesis, our prime interest is in transaction activities and its analysis for suspicious behaviour. Transactions on bitcoin blockchain are fairly more complex in depth but since the scope of this thesis is to analyse the bitcoin transactional data this thesis sheds light on the analytical perspective of bitcoin data. Chapter 5 discuss the anatomy of transaction activities regarding elements of data and structure.

2.5 Anomaly detection

Zimek and Schubert (2017) explain anomaly detection as the identification of unique items, observations or events that elevate suspicion by being significantly different from the majority of the data. Usually, anomalous items are interpreted as kind of abnormalities such as credit card fraud, structural defect or error in the text. Anomalies can also be depicted as noise, outliers and novelties. However, Pedregosa et al. (2011) explain that there is a significant difference between outlier detection and novelty detection. In the context of novelty detection, training data is not polluted by outliers, and we are interested in detecting whether a newly introduced observation is an abnormality for the system or not. Whereas, in the case of outlier detection, training data contains outliers which are defined as observations that are far from the others.

This thesis primarily focuses on detecting anomalies in blockchain data structure. Regardless of the domain of blockchain one can apply specific techniques to this data structure and attempt to identify anomalies. This is an unsupervised learning problem and data for anomaly detection cases is usually highly imbalanced. Due to data availability and academic literature, this thesis intends to make use of a public financial blockchain known as bitcoin. Bitcoin's transactional data goes through an intense pre-processing phase, and later various modelling methods are applied in order to identify transactions that are related to theft, heists or money laundering. A comprehensive evaluation process is performed that seeks to outline the optimal solution after comparing models. Studies in similar context have been carried out recently, however this thesis endeavours to approach the blockchain anomaly detection problem in a generalised way. Moreover, it also offers use of cutting-edge machine learning algorithms along with graph theory to analyse anomalies within blockchain data in an innovative manner.

2.6 Cyber fraud as a problem

Computer-oriented crimes have grown dramatically in the last past few decades. Making digital security highly essential in all domains. Sandle and Char (2014) issued a report estimating 445 billion USD annual damage to the global economy. According to Smith (2015) cybercrime costs can go as high as 2.1 trillion by 2019. North-Denver-News (2015) stated that in 2012 approximately 1.2 billion USD were lost to financial credit card fraud in the US. European-Central-Bank (2014) report reveals that 1 EUR in every 2,635 EUR on debit and credit card is lost to fraud. Al-

though cybercrimes can be further categorised into cyberterrorism, cyberextortion, cyberwarfare and financial fraud crimes; The prime focus of this thesis is in light of financial area.

2.6.1 Financial cyber frauds

The financial sector is one of the most affected domains by cyber crimes. The vulnerabilities of users and sometimes systems are the primary reasons for security breaches and hacks. Although the arrival of the blockchain has mitigated several risks dramatically, however, the majority of financial systems are hesitant of the upgrade due to internal policies and bureaucracy. Financial industry suffers enormous losses every year. Credit card frauds are the one the most common type of cyber fraud that takes places all over the world. Artificial intelligence plays a vital role in detecting credit card fraud, but still a significant amount of hackers are never caught. Krebs (2013) explains that around 40 million sets of payment card data were compromised in the hack on Adobe Systems. The information compromised included encrypted card payment number, customer names, card expiration dates and other order related information of customers. McCurry (2016) states in The Guardian that a coordinated attack was carried out by 100 individuals who used the data of 1600 South African credit cards to steal 12.7 million USD from 1400 stores within 3 hours. They also managed to flee from Japan before the authorities even discovered the heist. Even though many countermeasures are taken to make credit and debit cards secure, such cases depict that centralised systems are still vulnerable, ultimately exposing the whole financial industry to risk.

2.6.2 Blockchain thefts and heists

Blockchains bring privacy and security to the architecture of finance. However, certain people have been successful in fooling this in theory unbreakable infrastructure. The purpose of these hackers is usually to carry out an illegal activity without getting noticed. In order to do so, these hackers have to either tint their tracks or completely deceive the system in believing that their activities are legit. Several small to medium scaled cases usually never get reported. However, some big ones can make headlines. Bitcoin, being the first and oldest financial blockchain has also encountered many illegal activity challenges. Reid and Harrigan (2011) describe a Bitcoin theft known as 'All In Vain' in which around 25,000 bitcoins were stolen. Figure 2.6 is a visual representation of the theft patterns.

The red node in the graph above represents the hacker and green node victim.

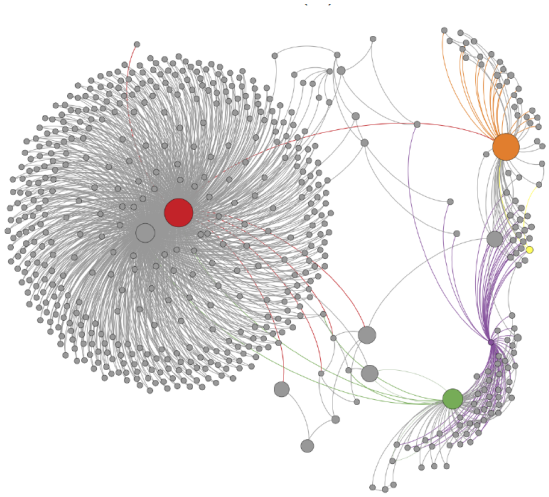

Figure 2.6. *All in vain robbery network; Reid and Harrigan (2011)*

A single bitcoin was stolen at the beginning which later led to the actual heist of 25,000 bitcoins. As seen in Figure 2.6, the hacker tried to hide his illegal activity by tainting the bitcoins using several small transactions. Moreover, a victim with the alias Stone-Man (2010) has written on bitcoin forum about his loss. 8999 bitcoins were robbed from him using his original private key. The victim in this case initially bought 9000 bitcoins from an exchange. Later on, he transferred these those to a disc and also backed up them in a USB flash drive. He also sent a single bitcoin to himself on another address for some unknown reason. After confirmation and backing up all the data of his wallet he later realised that there is an unrecognised transaction of 8999 bitcoins to an unknown address that he did not approve. Figure 2.7 illustrates how without his consent the bitcoins were hacked and sent to another address.

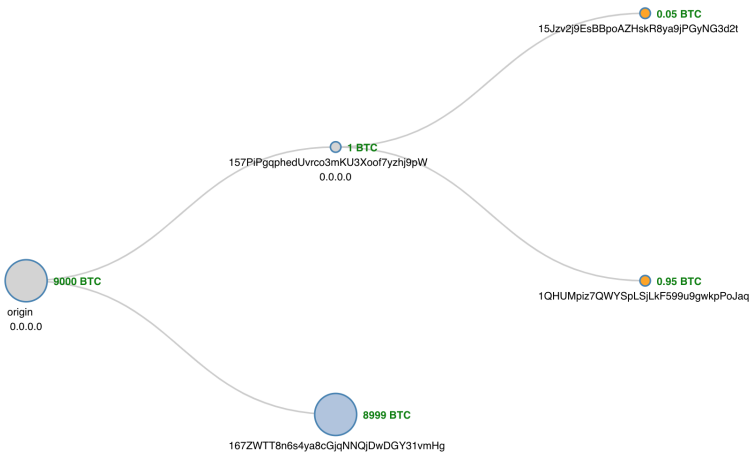

Figure 2.7. *Stone man loss robbery; Blockchain.com (2018)*

These examples illustrate some of the patterns that can be suspicious. Hence, we can mark similar activities and patterns as anomalies and attempt to create a model that can detect them. Bitcoin is also notorious about its use for illegal activities on the dark web such has involvement with money laundering, drugs and weapons. Individuals have exploited the technology as a payment system on the dark web to buy and sell illegal items and services. Christin (2013) explains how an online black market 'Silk Road' founded in 2011 and based on the dark internet was used to sell illegal drugs. Its monthly revenue was about 1.2 million USD. In October 2013 the USA Federal Bureau of Investigation shut down the website and arrested the person behind. Effects of these illegal activities damage not only the social credibility of the blockchain but also affects its value. Figure 2.8 shows a downfall in the market of bitcoin when the silk road seizure happened.

Figure 2.8. *Effect of the silk road seizure; Wikimedia-Commons (2018)*

3 Related work

Anomaly detection is broadly used in a wide area of applications such as fault detection, intrusion detection, fraud detection along with many others. The primary focus of this thesis is on fraud detection. Fraud detection is a well-studied area and can be partitioned into several sectors such as credit card fraud, virus intrusion, insurance fraud and many more. These researches usually utilise a variety of techniques to perform their analysis such as machine learning algorithms and network analysis methods. Recent development in the industry and previous research by Bolton, Hand, et al. (2001) shows that unsupervised machine learning algorithms behave in a more focused manner toward anomalies. However according to Phua et al. (2010) majority of fraud detection studies employs supervised machine learning techniques and focus on developing a complex model to learn the patterns of anomalies within the training data. Since applications of distributed ledger technologies are relatively new, the fraud detection research in this area is still ongoing. Blockchain technology is based on decentralized networks. It is observed from previous studies that in the majority of blockchain research use-cases, researchers either treat the problem from a network data perspective or crude data perspective. This chapter explains how previous researches have tackled blockchain anomaly detection problem and what approaches and outcomes they were able deduce.

3.1 Network based studies

B. Huang et al. (2017) approached the problem from a network behavioural pattern detection perspective. The data used in this study is not made public. According to the authors, this technique applies to every blockchain due to its generalised nature. However, the core idea behind is to find the behavioural patterns in the blockchain network and categorise them using newly introduced Behavior Pattern Clustering (BPC) technique. Transaction amounts changing over time are extracted as sequences to be clustered into several categories. These sequences are measured using similarity measuring algorithms; this study talks about Euclidean distance, Morse and Patel (2007) Dynamic Time Warping (DTW), Chen, Özsu, and Oria (2005) Edit Distance on Real sequence (EDR) and Longest Common Sub-Sequences (LCSS) by Vlachos, Kollios, and Gunopulos (2002). Authors of this study conducted tests and selected DTW as a sequence measure and dropped EDR and LCSS because blockchain data is noiseless and they focus on handling noisy data. A customised

version of K-means is introduced to detect patterns in the network which is named Behaviour Pattern Clustering (BPC), and results of BPC are compared with Hierarchical Clustering Method (HIC) and Density Based Method (DBSCAN). Final results showed that the BPC algorithm is more effective than existing methods. This study approached the blockchain network anomaly problem from an algorithmic aspect.

Signorini et al. (2018) focused on blockchain anomaly detection from a more networking nodes viewpoint. This research focuses on detecting and eliminating eclipse attacks. As explained by Heilman et al. (2015), eclipse attacks target single user nodes of the network instead of attacking the network as a whole. The attacker can hijack victims incoming and outgoing communication stream and inject malicious code in the system. Agenda of such attacks is usually to take over the control of the complete network which can cause severe damage to all users. The essence of this research is to contribute a decentralised system that can make use of all information collected from previous forks to protect the system against anomalous activities. Evolution of forks in a blockchain are prone to malicious activities and since eclipse attack happens on a single node rest of the network never gets informed about it. Signorini et al. (2018) proposed to create a blacklist that can inform other peers of the network about the malicious activity. A thread database is maintained to accumulate all known attacks which are later used to detect anomalies in the network. A toy network was set up to perform experimentation of the proposed solution, and it performed positively. Machine learning techniques are considered to be added in future studies which can enable this research to create a prediction model that can detect heterogeneous malicious transactions.

Pham and Lee (2016b) has proposed to approach Bitcoin blockchain anomaly detection from a network perspective. Based on Reid and Harrigan (2013), they converted the Bitcoin data into a network like structure, which is further divided into two primary graphs, user graph and transaction graph. The extracted user graph is used to attempt detection of suspicious users, while the transaction graph is used to attempt the discovery of suspicious transactions. Utilising both graphs, they not only try to detect abnormal users and activities but also establish a link among both. This results in a system that can also associate suspicious users with unusual transactions. Metadata of both graphs is extracted to create a new dataset which they have not made public. This metadata contains features such as in-degree, out-degree, balance amount of graph nodes and many other vital variables. Emerging dataset is quite large and includes all data of bitcoin blockchain from its creation to April 7th, 2013.

6,336,769 users along with 37,450,461 transactions were processed. Methods used for network anomaly analysis in the bitcoin network were K-means clustering motivated by Othman et al. (2014), Power Degree & Densification Laws inspired by Leskovec, Kleinberg, and Faloutsos (2007) and Local Outlier Factor (LOF) as mentioned by Breunig et al. (2000). K-means clustering is used in combination with Local Outlier Factor (LOF). To narrow down the list of potential k-nearest-neighbours in LOF, indices are calculated from K-means clustering. List of nodes for each cluster is obtained from k-means, and later on, a k-nearest-neighbour search is carried out to save computational time. Due to the computational limitations of experiments, only a small subset of extracted features were processed. One case of the anomaly was successfully detected out of 30 known cases. Challenges faced by the researchers in this study were mostly related to performing evaluation and validation. In reference to this study, LOF seemed to perform better than others at the network data of bitcoin. Moreover, this research does not talk about the false positive and true negative cases.

Zambre and Shah (2013) published a report that analysed bitcoin network dataset for frauds. Bitcoin network data analysed by this report was limited, containing network data only up to July 13, 2011. This report focused on detecting rogue users of the network who could potentially exploit network; dataset contained three known rogue users and 628 known victims. Provided graph-based data was used to extract more features out of the actual dataset, and a small subset of extracted features was selected to perform the final analysis. K-means algorithm was selected to create clusters of rogue and good users. On a high-level, this report was able to identify the rogue users but still was not able to create a clear separation between rogue and good users. Due to the lack of quality data, synthetic data was also generated using resemblance from robbery patterns. Analysis of synthetic data resulted in 76.5% accuracy rate, but non-synthetic data performed quite poorly.

3.2 Crude data based studies

Pham and Lee (2016a) have also published another study in which they researched anomaly detection in the bitcoin data using unsupervised learning methods. This study used the same dataset as their other study but approached the problem from a machine learning based perspective. The dataset comprised of graph metadata from the bitcoin user graph and transaction graph is pre-processed to extract a subset of features. Unsupervised machine learning algorithms chosen for this study were K-means clustering, Mahalanobis distance based method and Unsupervised Support

Vector Machines (SVM). The study was able to detect two known cases of theft and one known case of loss out of 30 known cases. Moreover, it does not address the false positive and true negative cases.

The research of P. Monamo, Marivate, and Bheki Twala (2016) took a slightly different approach to the bitcoin fraud detection problem. Bitcoin dataset provided by Brugere (2013) from Laboratory for Computational Biology at the University of Illinois is used in this research and is now unavailable due to unforeseen reasons. This research aimed to use K-means clustering and a variant of K-mean clustering proposed by Cuesta-Albertos, Gordaliza, Matrán, et al. (1997) in a multivariant setup to detect suspicious users and their related anomalous transactional activity. According to the authors, K-means clustering is capable of grouping similar instances together but lacks the prowess to detect anomalies, and Local outlier factor (LOF) is popular with outlier detection however it cannot be scaled for large datasets due to its computational complexity. This paper aimed to purpose an approach that could overcome the weakness of the mentioned algorithms. This study was able to perform comparatively better and was able to detect five fraudulent users out of 30.

A multifaceted approach was taken by P. M. Monamo, Marivate, and Bhesipho Twala (2016) to detect global and local outlier in the bitcoin network. Dataset used in this study is same as above mentioned studies and was also taken from Laboratory for Computational Biology at the University of Illinois by Brugere (2013). Data points farther away from the centroids of clusters regarding euclidian distance were considered Global outliers and were analysed using trimmed K-means according to Cuesta-Albertos, Gordaliza, Matrán, et al. (1997). Whereas by using kd-trees by Bentley (1975) the data points which had a significant distance from the nearest predetermined number of neighbours were considered local outliers. Top 1% of selective anomalous data was labelled as anomalies and the remainder was labelled as normal instances. Both supervised and unsupervised techniques were used in this study to analyse the network data. This study followed a pipeline in which data was pre-processed and labelled as normal or anomalous and then contextual binary classification techniques were applied to extract final analysis. The algorithms used for supervised classification were maximum-likelihood, logistic regression by Sarkar, Midi, and Rana (2011), boosted logistic regression by Friedman, Hastie, Tibshirani, et al. (2000) and random forest as mentioned by Breiman (2001). Thirty known fraudulent activities were evaluated against the analysis and unsupervised approach with K-means and kd-trees worked in collaborative agreement to detect anomalies. Five global anomalies were able to be identified by using K-means whereas in local

outliers kd-trees was able to identify 7 of the known thefts using. The two algorithms remained in agreement 22% of the time. In supervised learning, 70% of data was selected as training set while rest as the test set and 10-fold cross-validation was performed for evaluation for all algorithms. Results showed that random forests perform better as compared to others and were able to detect eight known thefts irrespective of class asymmetries. Moreover, this study also reveals important features that were able to learn patterns of anomalous activity in this particular study and also that local outlier detection works more efficiently as compared to global.

Hirshman, Y. Huang, and Macke (2013) aimed to make the exploration of bitcoin data easier, attempting to detect money laundering over bitcoin network using mixing services and tracking the outputs of mixing services back to their inputs. Whereas mixing services pools in money from different sources and mix them with intention to confuse the trail back to the funds original source. This study also uses the same bitcoin network dataset created by Brugere (2013) at the University of Illinois. Pre-processing techniques by Reid and Harrigan (2013) were used to associate public keys to users in order to establish a logical relationship within data. K-mean by James MacQueen et al. (1967) was used in combination Role Extraction (RolX)algorithm by Henderson et al. (2012) in an unsupervised manner for analysis. This study takes a unique approach of restructuring the dataset to uncover the laundering anomalies. It introduced a concept of hubs in which users with a high number of transactions were filtered to make dataset more effective for unsupervised algorithms. However, the result statistics were not shared in this research.

Patil et al. (2018) utilised a data mining approach to detect fraud in the bitcoin blockchain. Transactional data is used in this research to detect anomalies within the blockchain; It is an assumption that the majority of the network transactions are legitimate and only 1% of total data is fraudulent. The proposed system intends to scan the internal data with provided feature variables to filter out the fraudulent transactions. Data goes under pre-processing to eliminate the missing values, and a trimmed version of K-means clustering technique is utilised to categorise data into two clusters that represent fraudulent and legal transactions. As there is no labelled data available, it is very challenging to validate the results of the model for flagged suspicious activities. However, the research shows a promising result with improved detection rate with respect to previous studies. The future scope of research intends to do experimentation with under and oversampling of data as well as looking into the possibility of analysing data with graph-based algorithms and supervised machine learning algorithms.

In addition to heterogeneous data analysis, visual analysis has also attained popularity. Bogner (2017) seeks to understand and analyse anomaly detection in blockchains using visualised features. Their research not only proposed a theoretical approach but also practically implemented a solution to support it. The proposed solution runs a decentralized application on top of Ethereum blockchain and collects statistics from the blockchain to store it in a NoSQL database which later provides customizable visualisations and dashboards. A sophisticated machine learning core operates over the stored data to create visual analysis and detect anomalies online, and the results are represented visually on dashboards. Such an approach optimises interpretability and visualizability of machine learning algorithms. Studies on anomaly detection in the bitcoin have mostly focused on the transaction structure of the bitcoin network. However, ethereum blockchain with its smart contracts produces a complex context for analysis hence this research complements to transaction graph analysis strategy of previous researchers.

4 Anomaly detection methods

Anomaly detection is categorized into three main types of learning methods; Supervised learning methods, Semi-Supervised learning methods and Unsupervised learning methods. Majority of real-world anomaly detection problems fall under unsupervised learning domain such as credit card fraud detection, network intrusion detection and our blockchain fraudulent transaction detection. This thesis deems supervised and semi-supervised learning methods out of the scope and aims to focus entierly on unsupervised learning methods.

Unsupervised learning is the most flexible type of learning, as it does not require any class labels for training. It scores the data based on its intrinsic attributes such as distance, density or more and attempts to distinguish the anomalies from the standard data based on estimation. As illustrated in Figure 4.1, the majority of unsupervised anomaly detection algorithms can be roughly categorized into the following main groups Chandola, Banerjee, and Kumar (2009). This illustration is updated and differs from the original publication.

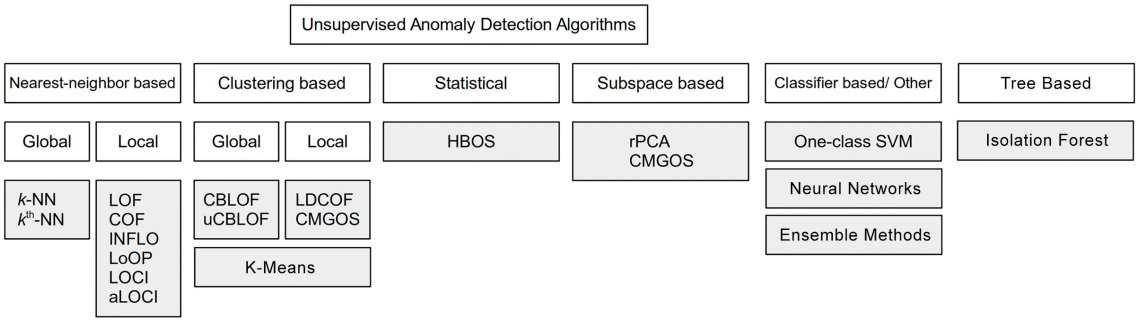

Figure 4.1. *A taxonomy of unsupervised anomaly detection algorithms that are used most in practice; Goldstein and Uchida (2016)*

This thesis selectively implements a set of unsupervised anomaly detection methods that can handle a large amount of data and can compute results in a reasonable time. Following sub-sections of this chapter will explain these algorithms in more depth.

4.1 Isolation Forest

Liu, Ting, and Zhou (2008) published an algorithm called "isolation forest" for unsupervised anomaly detection that achieved much popularity in recent years. The idea behind isolation forest is that it is comparatively more straightforward to isolate

anomalies from the data than to build a model that could estimate normal behaviour. In order to isolate an anomalous observation, the dataset is randomly and recursively partitioned until the only data point left in the partition is the observation. A tree structure is used to represent the recursive partition. The base of the isolation forest algorithm is on the ensemble of isolation trees, A forest of random isolation trees is constructed to estimate the measure of normality and abnormality of observations in the dataset.

An isolation forest is a combination of isolation trees, and these trees are actual binary trees in which each node has either zero or two child nodes. Let N be a node which is either a leaf node with no children or parent node with two child nodes N_l and N_r. To decide which children nodes go to which parent node a test is associated with node N. This test involves selecting a random feature q from the data and a random split point p such that the nodes $p < q$ are under N_l and $p \geq q$ are under N_r.

Multiple random trees are formed recursively, and data is partitioned in each one of them unless all observations are isolated for each iteration. The path length from the root node of the tree to the leaf defines the number of partitions required to isolate that observation. It is observed that for randomly partitioned data, the path length of anomalies is shorter than of standard observations. One of the reasons for this phenomenon to occur is that usually the number of anomalies in a dataset is smaller than the number of standard observations, which results in a shorter number of partitions and hence makes isolation of anomalous observation easier. Another reason for anomalous observations to be separated in earlier partitioning is that they have distinct attribute values as compared with standard observations. Figure 4.2 illustrates that anomalies are more sensitive to isolation. The figure below depicts the anomalous observation x_o requires only four partitions to be isolated, whereas isolating standard observation x_i requires twelve random partitions.

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$ containing n observations, random recursive isolation trees are built until each tree reaches the hight limit of $|X| = 1$ or all data in X have the same value. Presuming majority of data in X is distinct, and the isolation tree is fully grown, each observation is isolated to a leaf node. Then the tree will have n leaves and $n\text{-}1$ internal nodes, in total $2n\text{-}1$ nodes. The path length of an observation x can be calculated by heuristic $h(x)$ which is a measurement of the count of edges it takes to traverse from the root node to the leaf node. In order to use the isolation forest as an anomaly detection technique, it has to generate comparable anomaly scores as $h(x)$ cannot be directly used as an anomaly score. It

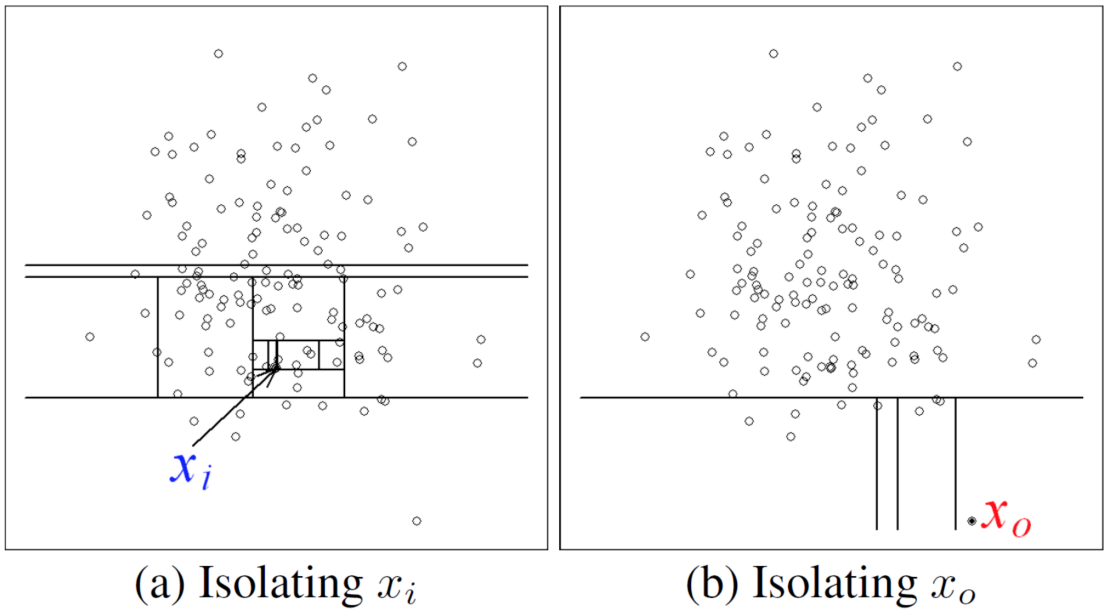

Figure 4.2. *The visual difference between the isolation of a standard observation x_i versus an anomalous observation x_o; Liu, Ting, and Zhou (2008)*

is because while the maximum height of a tree grows in the order of n, the average height grows in the order of $log\ n$. Isolation trees have a similar structure as binary search trees (BSTs). Therefore the average path length of a failed search in a binary search tree is comparable to the average of $h(x)$. As stated by Preiss (2008), the average path length of a failed BST search in a tree with n nodes can be calculated as

$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{4.1}$$

where $H(i) \approx ln(i) + 0.5772156649$ (Eulers constant). The anomaly score s for an observation x is defined as:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{4.2}$$

Here $E(h(x))$ is the mean of all the $h(x)$ from all isolation trees. While training an isolation forest algorithm, isolation trees are randomly created as explained above. Training requires two parameters: the number of trees t and the sub-sampling size. Tree height limit l is calculated based on the sub-sampling size. The sub-sampling process controls the data size for training and is an essential variable as it can directly affect the performance of the classifier. As a different sub-sample is picked to create an isolation tree each time, it can help isolate the anomalies in a better way. As for

anomaly prediction from the trained model, Equation (4.2) is used to calculate an anomaly score s as explained above. This anomaly score s ranges between the values from [0,1] where values being more close to 1 are more likely to be anomalies.

4.2 Histogram-based Outlier Score (HBOS)

Goldstein and Dengel (2012) introduced an unsupervised algorithm for anomaly detection. Histogram-based Outlier Score (HBOS) is a statistics-based method designed to handle large datasets. Its computation complexity is an attractive attribute for researchers.

A univariant histogram is created for each data feature. For categorical data frequency of values for each category is calculated, and the height of the histogram is computed. For numerical features, either static bin-width histograms technique or dynamic bin-width histogram technique is used. The static bin-width method uses k equal width bins over a value range, and the height of the bin is estimated by counting the samples which fall in each bin. The dynamic bin-width is determined by sorting all data points and then grouping a fixed number N/k of consecutive values, where N represents the number of total observations k represents the number of bins. Both static and dynamic approaches are introduced in this algorithm because real-world data have very different feature distributions, especially when features have a significant range of gaps in between. A fixed bin may estimate the density of anomalies poorly resulting in adding most of the data to a few bins. Anomaly detection problems usually have considerable gaps in the data range as most outliers are far away from standard observations. The authors mostly prefer the dynamic method if data distributions are unknown. The technique used to determine the value of k bins is the square root of instances N.

A particular histogram is computed for each dimension d, and density is estimated for the height of each bin. Normalization is applied over histograms so that their maximum height is 1.0. By doing so each feature is ensured to be weighted equally for anomaly score. HBOS for each observation p is computed using equation (4.3)

$$HBOS(p) = \sum_{i=0}^{d} \log\left(\frac{1}{hist_i(p)}\right) \tag{4.3}$$

An inverse multiplication of the estimated densities defines the score, assuming independence of features in similar manner to Kim et al. (2004). This score can also be represented as an inverse of discrete Naive Bayes where instead of multiplication

a sum of logarithms is used, using the fact that $log(a \cdot b) = log(a) + log(b)$. The reason for this trick is to make the HBOS algorithm less susceptible to errors due to floating-point precision, which can cause very high anomaly scores for extremely unbalanced distributions.

4.3 Cluster-Based Local Outlier Factor (CBLOF)

He, Xu, and Deng (2003) proposed an algorithm named cluster-based local outlier factor (CBLOF) that has properties of a cluster-based algorithm as well as the Breunig et al. (2000) local outlier factor (LOF) algorithm. The idea of anomaly detection in this algorithm revolves around clustering the data which are later used to compute an anomaly score in a similar way to LOF. Any arbitrary clustering algorithm can be plugged in the clustering step. However, the quality of the clustering algorithm's results directly influences the quality of CBLOF's results.

The algorithm proceeds in a manner that it clusters a dataset D using an arbitrary clustering algorithm and assigns each observation in D to a cluster. Their respective sizes sort the clusters such that $|C_1| \geq |C_2| \geq \cdots \geq |C_k|$ where C_1, C_2, ... , C_k represent all clusters with k as the number of clusters. The intersection of any pair of clusters should be an empty set, while the union of all clusters should represent all the observations of dataset D. The next step requires searching a boundary index value which separates Small Clusters (SC) from the Large Clusters (LC). There are two ways to calculate this boundary. The first is given by Equation (4.4).

$$(|C_1| + |C_2| + \cdots + |C_b|) \geq |D| \cdot \alpha \tag{4.4}$$

An alternative way is to use equation (4.5).

$$(|C_b| / |C_{b+1}|) \geq \beta \tag{4.5}$$

α and β are user-defined parameters in the equations mentioned above. α is between the range of [0,1] and represents the data comprised by the Large Clusters (LC). β value should be set to greater than one and defines the lower bound on relative size among a pair of consecutive clusters. equation (4.4) represents the approach which separates bigger clusters from the smaller ones by adding the biggest clusters to LC until some defined part of data is covered. (4.5), on the other hand, implements the idea that a significant boundary location can be found between LC and SC among clusters which vary relatively in size. Concisely, it chooses a boundary between two consecutive clusters if the next cluster is β times smaller than the

previous one. If two equations result in a way that value is different for boundary b, the smallest value is selected.

After deciding the value of b, the sets LC and SC are created. LC contains all the clusters with the index equal or smaller to the b and can be represented with LC = C_i | i ≤ b. Whereas SC consists of the remaining clusters and is represented by SC = C_i | i > b. Finally, CBLOF scores are calculated for each observation in D using equation (4.6).

$$CBLOF(t) = \begin{cases} |C_i| \cdot \min\left(\text{dist}\left(t, C_j\right)\right) & : t \in C_i, C_i \in SC \text{ and } C_j \in LC \\ |C_i| \cdot \text{dist}\left(t, C_i\right) & : t \in C_i, C_i \in LC \end{cases} \tag{4.6}$$

Data points in the large clusters (LC) are estimated as normal observations, and data points in small clusters (SC) are estimated as anomalous observations. A specific central point is calculated to represent each cluster and the distance from this point to each observation in the large cluster defines its anomaly score. A data point further away from the cluster centre would have a higher chance of being an anomaly, hence resulting in a higher anomaly score. For the data points not residing inside large clusters, their anomaly score is calculated based on the nearest data points to a large cluster. Data points which are far from large clusters end up having even larger anomaly score. CBLOF computes anomaly score in a similar way than LOF with the significant difference that distances are calculated with clusters instead of specific points.

4.4 Principal Component Analysis (PCA)

Principal component analysis (PCA) is usually considered as a dimensionality reduction algorithm. It can manifest variance-covariance of dataset features to create new variables which are represented by functions of original variables called principal components. These principal components are distinct linear combinations of p number of random variables X_1, X_2, ... ,X_p. Principal components carry unique properties such as they are uncorrelated to each other, the variance of components decreases in descending order representing the first component with the highest variance and moving to lower variance with every next component. However, the sum of the total variation in the original features X_1, X_2, ... ,X_p is always equal to the total variation of all the principal components combined. Principal components are trivial to calculate by using eigen analysis of correlation or covariance matrix of data features.

Although PCA is usually used for dimensionality reduction, Shyu et al. (2003) proposed an approach to utilize this algorithm for anomaly detection. If all variables from the dataset follow the normal distribution, the principal components *(PCs)* we compute should be independently normally distributed with their variance equal to the eigenvalues which we may obtain from single value decomposition. With this assumption, the algorithm states that $\sum_{i=1}^{p} \frac{PC_i^2}{\lambda_i}$ follows the chi-square distribution with p degrees of freedom. To detect the anomalies using the PCA algorithm, all obvious outliers are removed using Mahalanobis distance; this step is carried out multiple times to remove any data points with very high Mahalanobis distance. With S as a covariance matrix, x_i as an observation of i^{th} feature in data and $\overline{x}$ as the mean of all features, Mahalanobis distance D is defined as equation (4.7).

$$D = \sqrt{(x_i - \overline{x})^T S^{-1} (x_i - \overline{x})} \tag{4.7}$$

Further on, principal component analysis is applied to the training dataset, and their corresponding principal components are computed for the test dataset. Respective major and minor anomaly scores are computed using equation (4.8) and equation (4.9).

$$AS_1 = \sum_{j=1}^{q} \frac{PC_j^2}{\lambda_j} \tag{4.8}$$

$$AS_2 = \sum_{j=p-r+1}^{p} \frac{PC_j^2}{\lambda_j} \tag{4.9}$$

Predictions are made using computed anomaly scores. If $AS_1 \geq c_1$ or $AS_2 \geq c_2$ the observation is considered as an anomaly. Whereas if $AS_1 < c_1$ or $AS_2 < c_2$ the observation is labelled as a normal data point, where c_1 and c_2 are outlier thresholds. This version of PCA detects two types of anomalies: dependency-oriented anomalies and extreme-value anomalies. Extreme-value anomalies are the data points which have extreme coordinates in comparison to the data pattern and are detected by using equation (4.8). Dependency-oriented anomalies are anomalies that are usually those data points that may have a peculiar correlation among their features and can be detected using equation (4.9).

4.5 K-means

In clustering algorithms, K-means is one of the most popular choices for anomaly detection. It was introduced as an unsupervised learning algorithm by J. MacQueen (1967). The algorithm aims to partition the data into k different clusters where each sample belongs to the cluster with the nearest mean. Each cluster is represented by a centroid c which is the sample mean of observations in that cluster. The similarity measure used when comparing observations to the cluster is squared Euclidean distance, which can be computed by using equation (4.10) where x_i is the observations and c_i is the centroids and n represents the number of observations.

$$d^2(\mathrm{x},\, \mathrm{c}) = \sum_{i=1}^{n} (x_i - c_i)^2 \tag{4.10}$$

The first step of the process in the K-means algorithm is to initialize the k centroids. Then each observation is iteratively compared with each centroid and assigned to it based on the clusters least sum of squares. These assigned observations ultimately form clusters around the specific centroids. This process is the same as assigning each observation to its nearest centroid by using Euclidean distance or squared Euclidean distance as the minimum value for both should be same. This step in the algorithm is also known as the *expectation* step.

After each observation is assigned to their nearest centroids and first clusters are formed, the centroids are updated based on the new sample mean for each cluster. Centroids can be any arbitrary data point in the cluster representing the mean of the cluster; it does not have to be any one of the observations. The arithmetic mean is also minimizing the sum of squares within the clusters. This centroid updating step is also known as the *maximization* step.

At the time of termination of the K-means algorithm, results with final centroids and cluster assignment of all observations are delivered. The algorithm terminates when centroids have converged, which means the value of centroids stops updating with more iterations. There is no guarantee that k-means reaches the global optimum before termination, so it is a common practice to run the algorithm several times on the data and choose a model with the lowest within-cluster sum of squared distances. As the mean of each cluster can be any arbitrary data point running the algorithm every time may produce different solutions that have converged differently, but could be somewhat similar. If the centroids are initialized near to the local optimum, the algorithm is likely to converge faster. K-means is a variant of a more generalized

clustering approach Expectation-Maximization (EM). EM algorithm has a more probabilistic approach to clustering and creates soft-clusters in contrary to hard-clusters created by K-means.

Many techniques are proposed to initialize the centroids for fast convergence and fast termination. However, Arthur and Vassilvitskii (2007) proposed an approach named *k-means++* in which the standard initialization of the centroids is done by uniformly selecting data points from the dataset. Arthur and Vassilvitskii (2007) proved in their original paper that standard *k-means++* outperforms standard K-means in convergence time and minimizing within-cluster sum of squared distance.

4.6 Deep Autoencoders

Artificial neural network algorithms are relatively old but have gained popularity recently due to significant breakthroughs in technology. An autoencoder is a particular type of artificial neural network that attempts to replicate its input as closely as possible to its output. Concisely, its an algorithm that can learn a way to regenerate what is fed to it. J. Zheng and Peng (2018) state in their research that autoencoders achieve that by determining the encoding of input data. It allows them to reconstruct the input data using a computed encoding. As figure 4.3 illustrates the schematic structure of an autoencoder can be broken into two parts encoder and decoder.

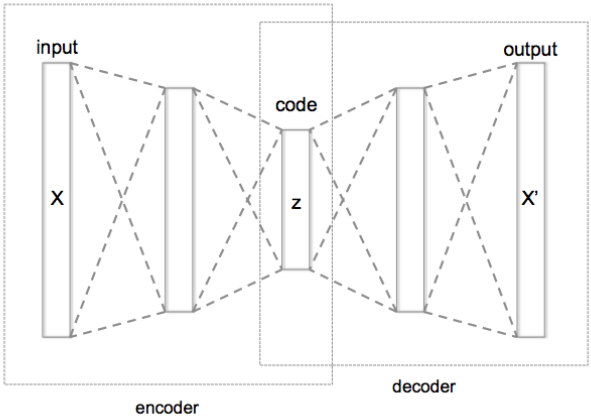

Figure 4.3. *Schematic structure of an autoencoder with 3 fully connected hidden layers; Commons (2016).*

The transition from the first layer X of the network to the middle layer z of the network represents encoder, and the transition from the middle layer of the network to the rightmost last layer X' of the network represents decoder. The first layer of

an autoencoder neural network is an input layer X, which may consist of m nodes, the same number of nodes as the dimensions of the data. The middle layer of the network z may have n nodes, where n represents the dimensions of encoding. The rightmost layer X' is the output layer and similar to the input layer shall have m nodes. As explained by Wang, Yao, and Zhao (2016) autoencoders are usually used for dimensionality reduction so that the dimensions of the learned encoding is usually smaller in size than the input.

As stated by the J. Zheng and Peng (2018) equation (4.11) represents the encoder e in the autoencoder and equation (4.12) represents the decoder d in the autoencoder. In the equations below W is an n x n matrix consisting of weights in the layers of neural network. b is the vector with size n which contains the bias of the layer and x is the input vector of length n. The σ represents a nonlinear transformation function such as sigmoid.

$$e = \sigma\left(W_{enc}x + b_{enc}\right) \tag{4.11}$$

$$d = \sigma\left(W_{dec}e + b_{dec}\right) \tag{4.12}$$

Deep autoencoders are neural networks with more complex structures such that each layer of encoder reduce the dimensions of the input to determine the hidden encoding behind the data. Vice versa each layer of decoder enlarges the dimensions until it reaches the input size. Autoencoder is an unsupervised learning algorithm that can encode and decode data while minimizing the error using the Hecht-Nielsen (1992) backpropagation algorithm. The reconstruction error is the metric used to quantify the similarity between the input and output of autoencoder.

To detect anomalies unlabeled data is given to the autoencoder for training, and it tries to minimize the reconstruction error over the training set. One of the most common reconstruction errors used is Lehmann and Casella (2006) mean squared error. Autoencoder tries to fit the model onto the normal dataset without anomalies, and since the autoencoder has not encountered any anomalies, the reconstruction should perform better on normal observations than on anomalies. Hence we can differentiate the anomalies from the standard data by calculating the reconstruction error. Typically the reconstruction error of an anomaly should be higher than the reconstruction error of normal observation. A threshold value α can be determined, which can set the boundary for classifying anomalous observations.

4.7 Ensemble Classification

Ensemble methods use a diverse combination of various algorithms to make a decision instead of relying on one. As mentioned by Raschka (2018) a standard method to combine the predictive power of these algorithms is called majority voting, which is a democratic solution to determine the final decision. Majority voting method can be represented by equation (4.13) where we predict $\hat{y}$ class label via majority voting of each classifier C_j.

$$\hat{y} = \text{mode}\left\{C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_j(\mathbf{x})\right\} \tag{4.13}$$

Another addition to the majority voting technique is that it also allows the association of weights with every algorithm making each algorithm's decision less or more effective in the final decision. Weighted majority voting can be illustrated by equation (4.14) here we can compute a weighted majority vote w_j by associating it with classifier C_j.

$$\hat{y} = \arg\max_i \sum_{j=1}^{m} w_j \chi_A \left(C_j(\mathbf{x}) = i\right) \tag{4.14}$$

Here χ_A is the characteristic function $\left[C_j(\mathbf{x}) = i \in A\right]$, and A is a unique set of class labels. Raschka (2018) explains another method called soft voting. Soft voting takes into account decision probabilities p of each classification algorithm to make a final decision and can be represented by equation (4.15) where w_j is the weight that can be assigned to the j^{th} classifier.

$$\hat{y} = \arg\max_i \sum_{j=1}^{m} w_j \chi_A \left(C_j(\mathbf{x}) = i\right) \tag{4.15}$$

5 Frameworks and dataset

This chapter dives into the technologies and frameworks used to conduct experiments[1] for this thesis and the life cycle of the blockchain anomaly detection process. Along with the exploratory analysis illustrations of the data it also explains how the dataset is constructed and pre-processed for the algorithms to model. This chapter also explains in detail the evaluation methods utilized for this thesis research.

5.1 Technology and Frameworks

The programming language used to conduct experiments for this thesis is *Python (version 3.6.7)*. Numerous libraries are used for different purposes to assist experiments. However, the major libraries are as follows:

- **Numpy (version 1.17.0):** NumPy is the fundamental package for array computing with Python and used in various computation operations.

- **Pandas (version 0.25.0):** It offer robust data structures for data analysis is and used for data handling and manipulation.

- **Scikit-Learn (version 0.20.3):** Implements various standard machine learning and model evaluation algorithms such as K-means, ROC Curve, F1-Score and more.

- **Matplotlib (version 3.1.1):** Is a plotting package used for visualization of data.

- **Seaborn (version 0.8.1):** Is a statistical data visualization tool used for data visualization.

- **Pickle (version 4.0):** This module implements binary protocols for serializing and de-serializing a Python object structure and is used for model preservation.

- **PyOD (version 0.7.2):** A Python toolkit for scalable outlier detection (Anomaly Detection) that implements the majority of anomaly detection algorithms such as HBOS, CBLOF, PCA, Isolation Forest and more.

[1] https://github.com/epicprojects/blockchain-anomaly-detection

• **Keras (version 2.1.6):** Deep learning library that makes implementing neural networks more comfortable to manage and understand.

• **mlxtend (version 0.17.0):** Machine learning library extensions used in this thesis for combining all models for voted ensemble classification.

• **NetworkX (version 2.1):** Python package for creating and manipulating graphs and networks used to generate the transaction graph from raw bitcoin data.

• **Hyperopt (version 0.1):** Distributed Asynchronous Hyperparameter Optimization library used for estimating the best parameters for anomaly detection algorithms.

• **imblearn (version 0.5.0):** Toolbox for the imbalanced dataset in machine learning used in this thesis for synthetic data creation.

• **bitcoin-blockchain-parser (version 0.1.4):** The library developed by Calvez (2015) provides a parser for the raw data stored by bitcoind; it was modified for this thesis and is utilized to read and extract data from raw bitcoin .blk files.

5.2 Dataset

Raw bitcoin blockchain data is used to create the dataset[2] for this thesis. All bitcoin data was synced from the internet using the bitcoin client software. All the data in bitcoin blockchain is stored into a public ledger and is represented by the currency unit called the Bitcoin (BTC). The ledger data contains all the bitcoin transactions from the beginning of the network creation to now. As recorded by Blockchain.com (2019) there are approximately 450,000,000 transactions within the bitcoin ledger. For each transaction, there can be multiple numbers of sender and receiver addresses. Moreover, a single user can own multiple addresses, and each user is anonymous as there is no personal information associated with any of the addresses.

The synced raw data from the bitcoin client was filtered out by the year parameter to extract a subset. This subset was extracted by writing a python snippet and represented the data from the year 2011 to 2013, which is approximately 29,000,000 transactions. The reason for selecting this specific range of years is the availability of

[2]https://ieee-dataport.org/open-access/bitcoin-transaction-network-metadata-2011-2013

anomalous bitcoin transactions data. Bitcoin Forum (2014) has a comprehensive collection of BTC fraud transactional activity; it has covered details about categories such as BTC Thefts, BTC Hacks, and BTC Losses, the red-flagged transaction of interest in each case along with details about the date and how much BTC was stolen or lost. Data from Bitcoin Forum (2014) was extracted to create an anomalies dataset[3] using a crawler written in python. This thesis used the following technique to tag malicious transaction while creating the final dataset: all the child transactions of a malicious transaction were also tagged as malicious. So, if someone stole some bitcoins and that particular transaction was tagged as a malicious activity, all the further occurring transactions such as transactions moving those stolen bitcoins elsewhere were also tagged as malicious.

Raw bitcoin data needed to be transformed into a meaningful structure and labeled using fraudulent transactional cases. The transactional data from the raw files were parsed to be represented as a directed graph structure. As illustrated by Reid and Harrigan (2013), the data is logically connected and can be represented as a transaction network. This transaction network T represents the flow of bitcoins between transaction throughout a period. Each vertex of this graph may represent a transaction, and each directed-edge between a source and a destination may represent details such as the number of bitcoins and timestamp of that transaction. The directed-edge is initiated from the input of the transaction directing towards the corresponding target output. Figure 5.1 represents a graphical illustration of a sub-graph from the transaction graph.

Figure 5.1 shows an example sub-network of network T. Transaction t_1 has one input and two outputs, it was added to the blockchain on 1^{st} May 2011. One output of t_1 is a transfer of 1.2 BTC (Bitcoins) to a user with public-key pk_1. The public keys are not illustrated in the figure 5.1 above. Transaction t_2 has two inputs and two outputs; it was carried out on 5^{th} May 2011. One of the outputs of t_2 is the transfer of 0.12 BTC to a user identified by public-key pk_2. t_3 transaction has two inputs and one output it was added to the ledger on 5^{th} May 2011, Inputs of t_3 are connected to outputs of t_1 and t_2 where t_3 itself have only one output t_4 that transfers 1.32 BTC on 5^{th} May 2011.

This transaction network graph is represented by a directed acyclic graph (DAG) and needs to be stored in memory in order to create the data set for anomaly detection. After experimenting with various storage techniques such as Relational

[3]https://ieee-dataport.org/open-access/bitcoin-hacked-transactions-2010-2013

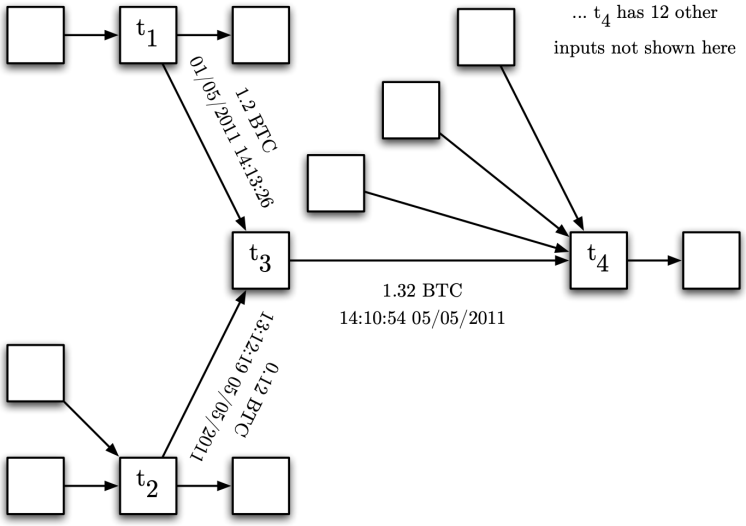

Figure 5.1. *An example sub-network from the transaction network. Each rectangular vertex represents a transaction, and each directed edge represents a flow of Bitcoins from an output of one transaction to an input of another; Reid and Harrigan (2013).*

Database, Graph Database, and Standard File System, it was determined that the fastest way to create a DAG was from a standard file system. A comma-separated value (CSV) dataset[4] was created with variables such as input-transaction hash, output-transaction hash, timestamp, and bitcoin value. Each line of this CSV file represented at what time and how many bitcoins (BTC) were transferred from the input-transaction hash to the output-transaction hash. This CSV file was later used to create an in-memory directed acyclic graph using python NetworkX library.

An in-memory directed acyclic graph (DAG) was created using the previously generated CSV file. In this DAG each vertex represented a transaction had a certain number of indegree and outdegree values along with a specific number of bitcoins (BTC) flowing-in and flowing-out. The DAG was traversed to extract the features and incorporate them into a dataset. The following features were extracted from the DAG to create a final dataset for anomaly detection.

- **In-degree:** Number of transactions from which a given transaction receives money.
- **Out-degree:** Number of transactions that a given transaction gives money to.
- **In-BTC:** Number of bitcoins on each incoming edge to a given transaction.

[4]https://ieee-dataport.org/open-access/bitcoin-transactions-data-2011-2013

- **Out-BTC:** Number of bitcoins on each outgoing edge from a given transaction.
- **Total-BTC:** Net number of bitcoins for a given transaction considering all in- and out-going edges from that transaction.
- **Average in-BTC:** Average number of bitcoins on each incoming edge to a given transaction.
- **Average out-BTC:** Average number of bitcoins on each outgoing edge from a given transaction.
- **Is-Anomaly:** Status of a given transaction if it is malicious or not.

While traversing the DAG, each transaction was also labeled as either anomalous or non-anomalous resulting in a final dataset consisting of metadata extracted from the DAG. The final dataset had around 30 million normal non-malicious transactions and 108 malicious transactions. It is to be noticed that the data is highly imbalanced, and it is challenging to create a robust model for anomaly detection. Not all anomaly detection algorithms are capable of handling such a massive amount of data, so for some algorithms, a specific number of sub-samples were modeled and compared to find the best fitting model. Also Synthetic Minority Over-sampling Technique (SMOTE) was used to perform over-sampling over the class of malicious transactions in order to have more training instances for modeling. However, only non-synthetic data was used to test the model.

5.3 Exploratory Data Analysis (EDA)

Exploratory analysis of the data includes various visualizations that help understand the data and the relationship among its dimensions in a more in-depth manner. It also helped in detecting class imbalance within data as well as it feature's correlation. The exploratory analysis also helped in understanding what kind of transformation can be applied on data to prepare it before modeling.

Exploratory data analysis revealed some insights into the data. As also mentioned in previous section, the class distribution of the dataset is highly imbalanced with 30,248,026 non-malicious data points and only 108 malicious transactions, make malicious transactions to be only approximately 0.00035% of whole data. A frequency histogram illustrates it in figure 5.2 where x-axis of the plot represents a non-malicious class with '0' and malicious class with '1' and the y-axis represents

the frequency of classes. The y-axis of the histogram was log-scaled for better visual representation.

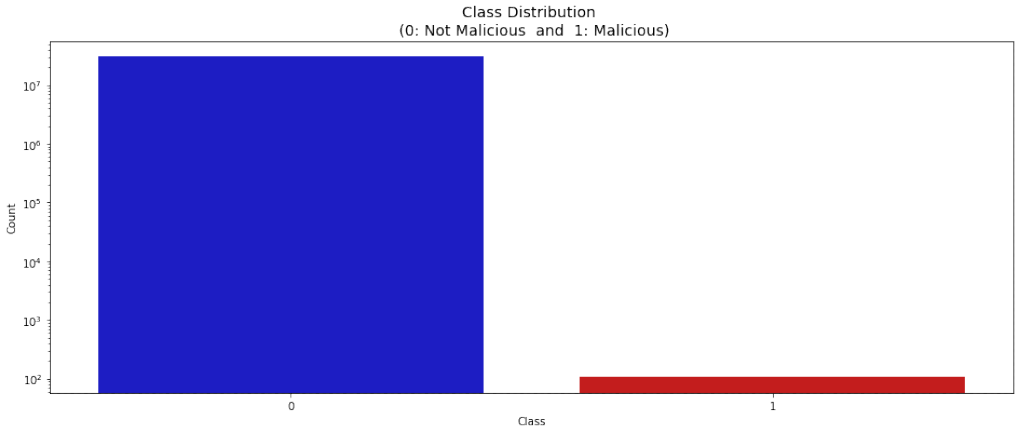

Figure 5.2. *A frequency histogram of the extracted bitcoin transactional data representing distribution of malicious and non-malicious transactions in the dataset.*

Distributions of all variables were plotted to have a visual sense of the data; e.g. if it is skewed and requires any transformation or normalization. Figure 5.3 illustrates the data in a grid of distribution histograms. Each histogram represents a single feature of the dataset and its respective data distribution. For the visual representation, both x-axis and y-axis were log-scaled. We can see in Figure 5.3 that all variables of the data highly follow a right-skewed distribution. This skewness of data can cause difficulty in the modeling process. Transformations to reduce the right-skewness of the data include square root, cube root, and log. According to this thesis's experiments, a Log function transformation seems to have a positive effect on the data. However, according to the table 5.1, our data contains of zero values, and standard log transformation cannot be applied to it. Table 5.1 also describes other statistics of the data such as mean, standard deviation, min value and max value for each feature of the dataset before the transformation.

	Indegree	**Outdegree**	**In-BTC**	**Out-BTC**	**Total-BTC**	**Mean-in-BTC**	**Mean-out-BTC**
mean	2.149	2.149	54.842	54.842	109.683	49.240	28.771
std	7.421	4.512	1303.303	1301.473	2602.527	972.921	712.557
min	0	0	0	0	0	0	0
max	1932	1322.000	550000	500020.7	1050000	499259.58	500000

Table 5.1. *Untransformed data statistics.*

Since standard log transformation cannot be applied to the data, on several experiments a *log(x+1)* transformation was applied along with *'Robust Scaler'* transformation. The purpose of the robust scaler transformation is to standardize the

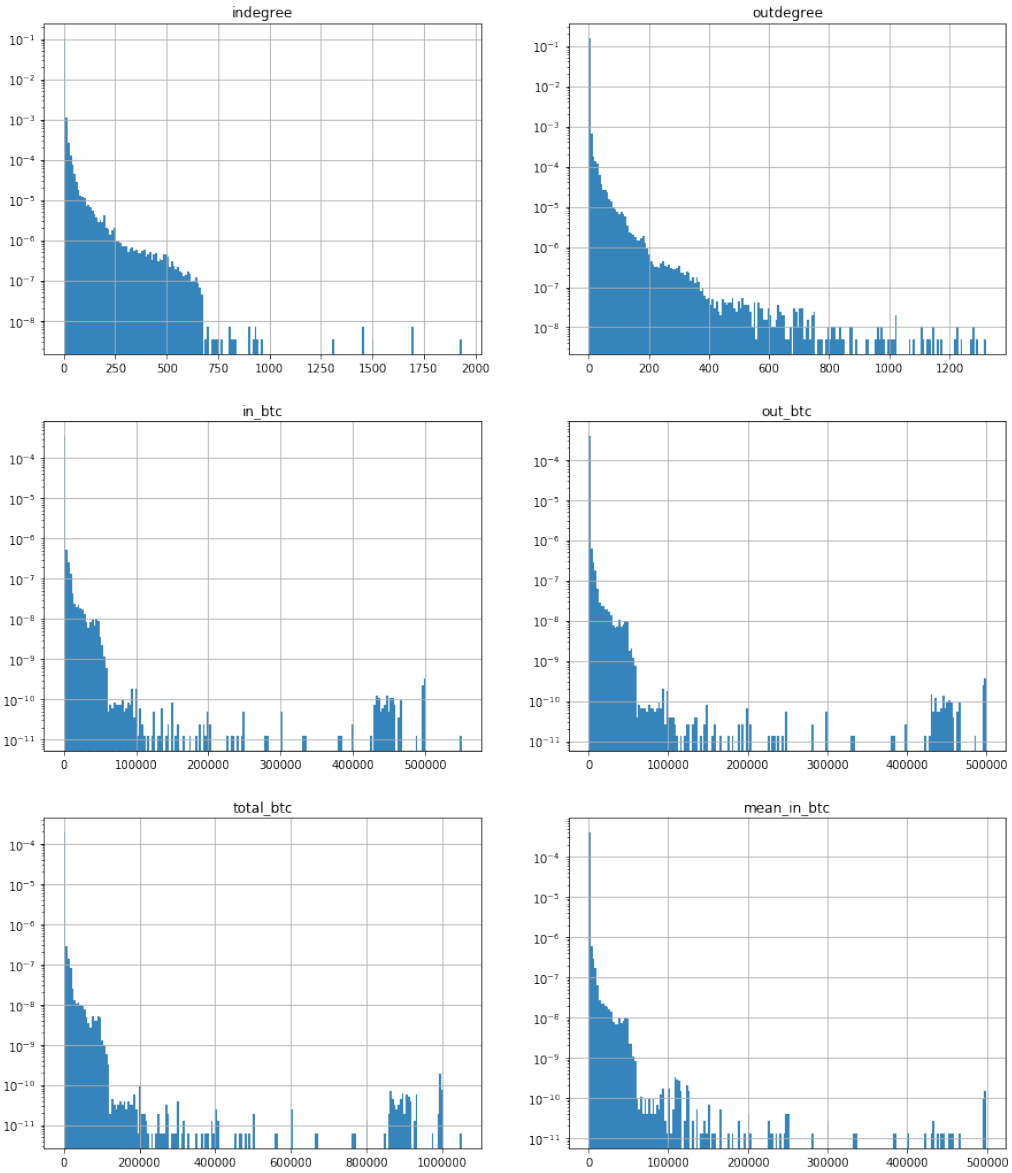

Figure 5.3. *A frequency histogram of the extracted bitcoin transactional data representing distribution of malicious and non-malicious transactions in the dataset.*

data by removing the median and scaling the data according to the Interquartile Range (IQR) between the 1st quartile (25th quantile) and the 3rd quartile (75th quantile). Scaling and standardization is applied to each feature independently and is achieved by computing relevant statistics. Standardization is a common requirement of several machine learning algorithms. Usually, it is achieved by removing the mean and scaling to the unit variance. However, in the case of outliers, sample mean and variance can be influenced negatively, hence affecting final results. In such cases, interquartile range and median can often give better results, and that is the motivation behind particularly choosing the robust scaler.

After the normalization and standardization transformations have been applied, the features of the data are observed to be less skewed than before. Moreover, the data is in a more reasonable range, and all the units of the variables have been standardized. This pre-processing step can help various machine learning estimators to perform better. However, the data is still skewed to some level and may cause complexity when modeled. Figure 5.4 illustrates the data distribution histograms after the transformation was applied. Table 5.2 describes statistics of the transformed data.

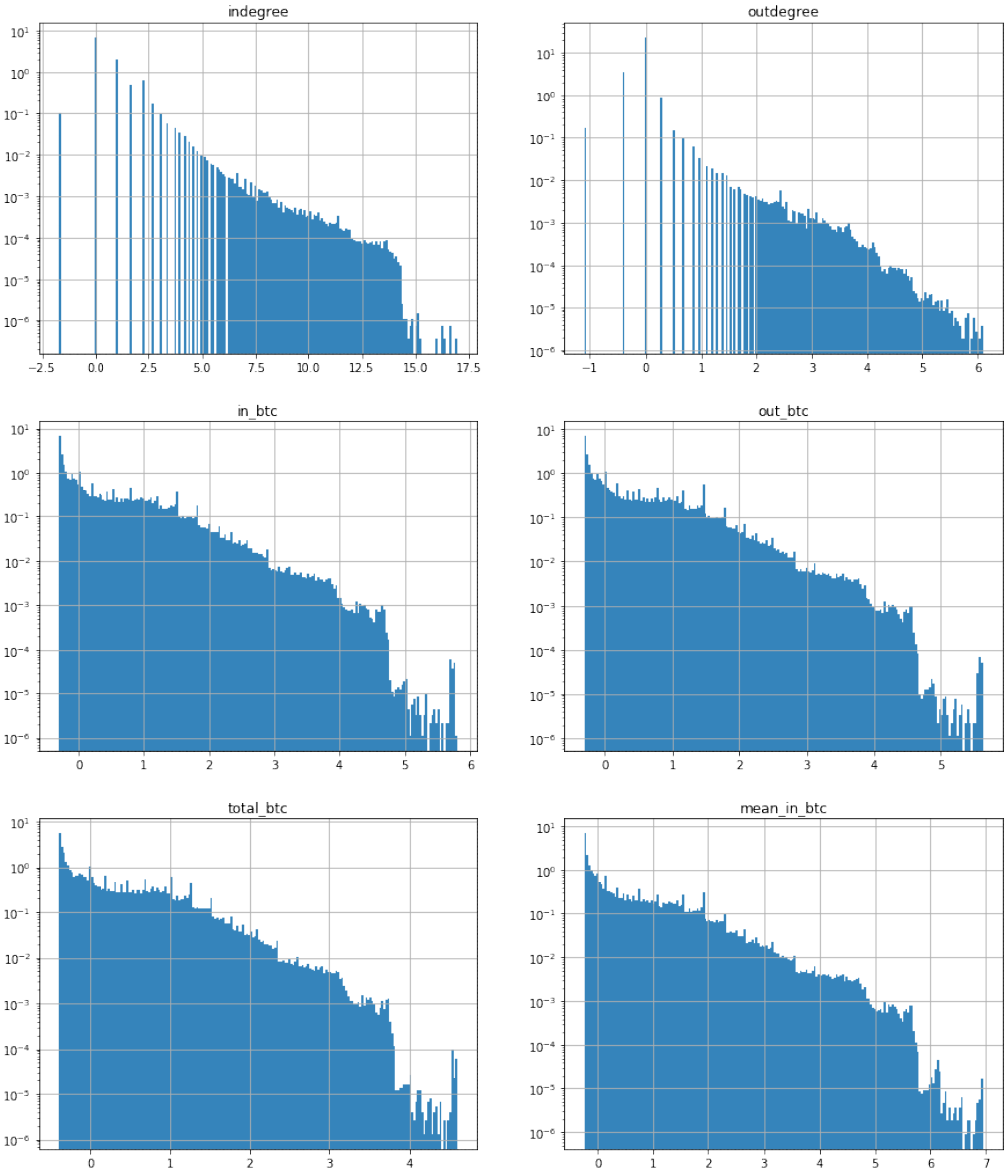

Figure 5.4. *Frequency distribution histograms of the transformed data.*

	Indegree	Outdegree	In-BTC	Out-BTC	Total-BTC	Mean-in-BTC	Mean-out-BTC
mean	0.613	-0.026	0.312	0.307	0.230	0.408	0.395
std	1.138	0.268	0.729	0.717	0.648	0.831	0.836
min	-1.710	-1.099	-0.304	-0.304	-0.391	-0.235	-0.240
max	16.953	6.089	5.799	5.623	4.599	6.946	7.668

Table 5.2. *Transformed data statistics.*

Correlation matrices are one of the essential factors of understanding data. They can aid us in determining if the features heavily influence a specific transaction to be malicious. Both extremely positive or negative correlation can reveal a feature's ability to affect maliciousness. To make sure that high data imbalance does not affect feature correlation, a second correlation matrix for a random subsample with balanced classes is also created and analyzed. Figure 5.5 visualizes the correlation matrix for complete data, whereas Figure 5.6 illustrates the correlation matrix for a balanced class random sub-sample. Based on both correlation matrices, we can say that `indegree` and `outdegree` exhibit negative correlation with maliciousness. The lower these correlation values are, the more likely the result will be a fraud transaction. On the contrary the features `in_btc`, `out_btc`, `mean_in_btc`, `mean_out_btc` and `total_btc` exhibit positive correlation. The higher these values are, the more likely the end result will be a fraud transaction. Strong blue color represent high correlation values, strong red color represent low correlation values, and all shades in between show values varying in between.

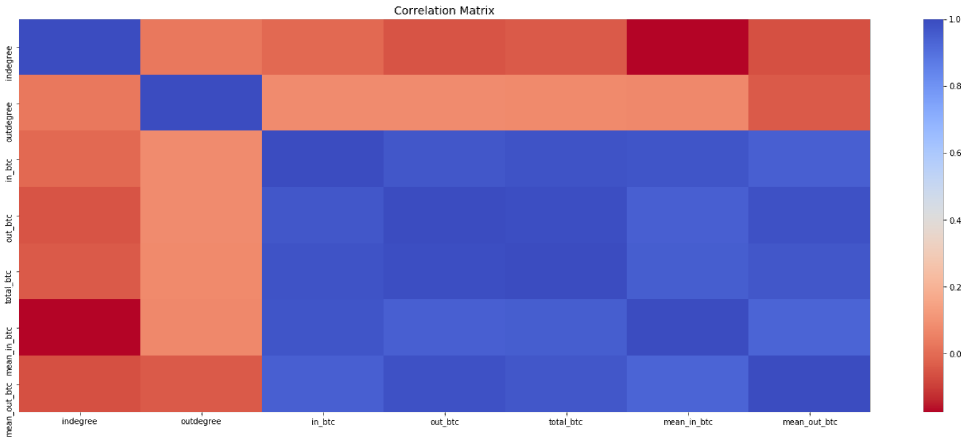

Figure 5.5. *Correlation matrix visualization of complete data.*

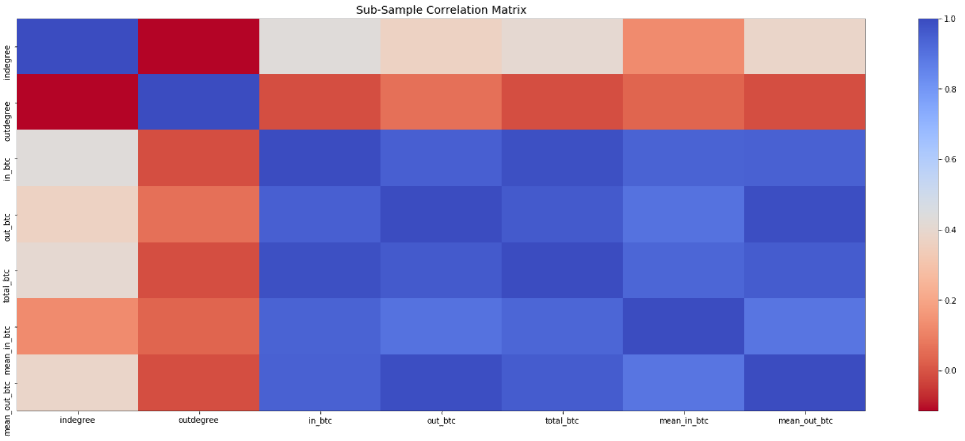

Figure 5.6. *Correlation matrix visualization of class balanced random sub-sample.*

Pair plots are a powerful tool that can help in data distribution exploration and uncovering relationships among data dimensions. It provides a comprehensive first glance at the data visually that can lead to further analysis decisions. Figure 5.7 shows a 7 x 7 grid structure in which each x-axis variable represents a relationship plot with each y-axis variable. The diagonal of this grid shows a histogram for each variable, thus depicting their distributions. The legend of the plot on middle-right helps in understanding the data-points. All blue data points in the pair plots represent normal non-malicious transactions whereas all red data points in the pair plots represent malicious transactions which are marked as fraudulent. As observed from Figure 5.7, there is a visible pattern of malicious transaction in-between several variable relationships. This pair plot is based on a complete data set consisting of approximately 30 million transactions. However, as mentioned in previous subsections, the data was normalized and standardized for better visual representation. This thesis aims to learn and estimate models of the patterns visible in the multiple pairwise bivariate distributions plot. Due to computation and storage scope limitations, previously known studies have not computed this many variables to extract a pattern. As seen in the illustration, the data is highly non-linear and would require a complex representation to learn.

This chapter also describes the implementation and results for all the research experiments for this thesis. It explains the metrics that are used to evaluate the results. Due to the nature of this research, multiple evaluation techniques were required in accordance with their respective algorithms and limitations.

In all experiments for this thesis, data is passed through a pipeline, which performs filtration, normalization, and scaling transformation over data to prepare it for

modeling. Although this experiment is treated as an unsupervised activity, the data is divided into a training set and a test set for evaluation purposes. The training set for isolation forest contains 80% of the data and consists of 24,198,425 non-malicious data points and 82 malicious data points. The test set contains 20% of the data and consists of 6,049,601 non-malicious data points and 26 malicious data points.

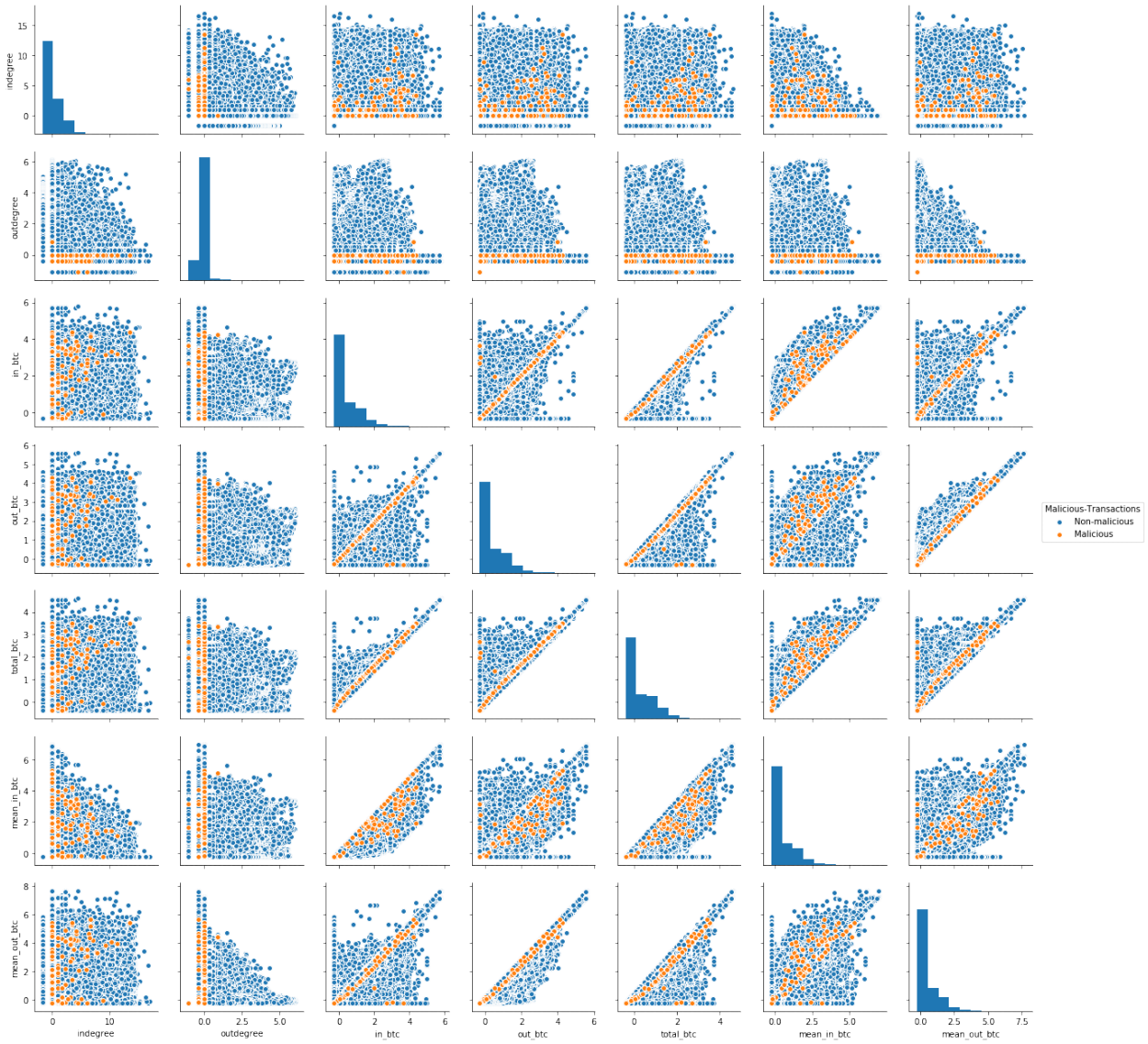

Figure 5.7. *Pair-plot grid between all feature variables visualizing the patterns of malicious and non-malicious transactions. The legend on middle-right describes the data point classes.*

5.4 Evaluation Metrics

This sub-section explains various evaluation metrics that are utilized in this thesis. Due to the high class-imbalance nature of the data in this thesis, conventional evaluation techniques are not applicable. However, various state-of-the-art evaluation metrics such as confusion matrix, recall, precision, accuracy, and more in combination a fair evaluation.

A confusion matrix C is a matrix that represents the error of a classification problem C_{ij} such that C_{ij} is equal to the number of observations belonging to group i but predicted to be in group j. In a binary classification scenario, the count of $C_{0,0}$, True Negatives (TN), represents observations that were negative and predicted negative. $C_{1,1}$, True Positives (TP), is the count of positive observations that were predicted positive. $C_{0,1}$, False Positives (FP), is the count of observations which were negative but predicted positive, and $C_{1,0}$, False Negatives (FN), is the count of negative observations predicted positive. Data from a confusion matrix can be used to compute various evaluation scores for a model, such as accuracy, precision, and recall.

Micro average metrics are sustainable to use when dataset classes vary in size and macro average metrics when it is vital to determine how the system performs overall all classes of data. For this thesis research, we want to determine the overall performance of the system across both malicious and non-malicious classes, so macro average metrics are primarily used. Micro average metrics in sum up True Positives (TP), False Positives (FP) and False Negatives (FN) for different classes individually to obtain statistics. Whereas macro average metrics are straight forward and compute the average of statistics, i.e., recall, precision on different classes of the data. Macro evaluation metrics are explained in more detail below.

Accuracy is an intuitive performance measure that computes the ratio between predicted observations and total observations and can be calculated by $\frac{TP+TN}{TP+FP+FN+TN}$. It gives a general idea about how well a model is trained, however for imbalanced datasets it does not perform well and can be deceiving. Balanced accuracy, on the other hand, is a more practical approach to calculate the accuracy of a model when data is imbalanced. Balanced accuracy is defined as the average of recall score R obtained on each class and can be defined as equation 5.1.

$$\text{balanced accuracy} = \frac{1}{C}\sum_{j=1}^{C} R_j \tag{5.1}$$

The ratio of correctly predicted positive observations to the total predicted positive observations is called precision and can be computed using $\frac{TP}{TP+FP}$. Whereas for C classes, the macro precision score can be described as equation 5.2 in which TP_j and FP_j represent true positives and false positives of each class label j.

$$\text{macro precision } = \frac{1}{C}\sum_{j=1}^{C}\frac{TP_j}{TP_j + FP_j} \tag{5.2}$$

Recall score is the ratio of correctly predicted positive observations to all the observations in the actual positive class and can be computed by $\frac{TP}{TP+FN}$. Macro recall score for C classes is defined as equation 5.3 in which TP_j represents true positives and FN_j represents false negatives for each class label j.

$$\text{macro recall } = \frac{1}{C}\sum_{j=1}^{C}\frac{TP_j}{TP_j + FN_j} \tag{5.3}$$

A weighted average of precision and recall is defined as F-Score, it can also be described as a harmonic mean of precision and recall and can be calculated by $\frac{2\times(Precision\times Recall)}{Precision+Recall}$. As equation 5.4 depicts, macro-f1 score is the harmonic mean between precision P_j and recall R_j.

$$\text{macro } F_1 = \frac{1}{C}\sum_{j=1}^{C}\frac{2\times P_j \times R_j}{P_j + R_j} \tag{5.4}$$

A receiver operating characteristic curve (ROC) is a visual illustration of diagnostic ability for a classifier as its discrimination threshold is varied. A ROC curve is created by plotting a graph between true positive rate (TPR) and false positive rate (FPR) at different threshold settings. True positive rate (TPR) is known as sensitivity or probability of detecting the right answer. Whereas the false positive rate (FPR) is known as fall-out or the probability of false-alarm of an algorithm. In a machine learning problem ROC curves play the role of figuring out how good the model is. The area under the curve (AUC) of a receiver operating characteristic curve (ROC) estimates a score that describes the model performance. Higher the AUC the better the performance of the model.

6 Research Results

This chapter explains the evaluation results for all algorithms and how they were obtained. Various evaluation metrics explained in section 5.4 were used to perform the assessment. Due to the vast amount of data, the time complexity for some algorithms was enormous. However, to tackle the problem, we randomly subsampled 1/10th of the training data and fit 20 different models with fine-tuned hyper-parameters. Parameters such as sub-sampling 1/10th of data and training exactly 20 models were calculated based on experiments and the values were finalized when the model's performance stopped changing even when the sample size or the number of models were increased.

6.1 Isolation Forest

To handle the extreme class imbalance, Synthetic Minority Over-sampling Technique (SMOTE) is utilized to generate more synthetic malicious training samples so that the algorithm can detect a pattern of anomalies within the data more efficiently. The ratio for newly generated synthetic data points is also estimated by hyper-parameter optimization. All hyper-parameters are optimized by minimizing a loss function. For this research case, the loss function chosen is the complement of the Macro F1 score. Optimized parameters estimated for isolation forest are 40 N-estimators and a 0.057 contamination/oversampling fraction. All trained models are evaluated using a single test set, which contains non-synthetic data unseen by all models. Table 6.1 describes the evaluation metrics for training data, and the table 6.2 describes evaluation metrics for test data.

Iteration	Accuracy	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	Macro-ROC	Precision	Recall	F1	ROC	Time
16	0.944288	0.733215	0.727564	0.733215	0.730347	0.877941	0.483863	0.496625	0.490161	0.877941	93.2479
4	0.942917	0.733721	0.721837	0.733721	0.727591	0.880167	0.47231	0.499235	0.4854	0.880167	72.8744
14	0.942915	0.733703	0.721827	0.733703	0.727577	0.873728	0.472292	0.499199	0.485373	0.873728	93.8357
19	0.942477	0.731424	0.719772	0.731424	0.725416	0.877053	0.468431	0.494856	0.481281	0.877053	92.8283
1	0.942361	0.73098	0.719247	0.73098	0.724928	0.869639	0.467428	0.494044	0.480368	0.869639	93.6913
5	0.942357	0.730625	0.719188	0.730625	0.72473	0.869266	0.46735	0.493297	0.479973	0.869266	72.6013
10	0.941957	0.729012	0.71737	0.729012	0.723007	0.874808	0.463887	0.490325	0.47674	0.874808	96.4204
15	0.941899	0.728732	0.717102	0.728732	0.722733	0.874072	0.46338	0.489796	0.476222	0.874072	93.914
7	0.94167	0.727586	0.716033	0.727586	0.721626	0.864868	0.461366	0.487621	0.47413	0.864868	93.7485
6	0.941509	0.726762	0.715279	0.726762	0.720839	0.875373	0.459948	0.486055	0.472641	0.875373	93.6141
2	0.941547	0.726146	0.715344	0.726146	0.720584	0.865821	0.460149	0.484706	0.472109	0.865821	72.668
17	0.941373	0.726154	0.714654	0.726154	0.720221	0.87482	0.458764	0.484916	0.471478	0.87482	93.1396
11	0.941423	0.72565	0.714784	0.72565	0.720054	0.870189	0.459083	0.483793	0.471114	0.870189	93.635
20	0.941322	0.725891	0.714414	0.725891	0.71997	0.856564	0.458313	0.484416	0.471003	0.856564	94.1311
12	0.941231	0.725456	0.713992	0.725456	0.719542	0.874049	0.457515	0.483597	0.470195	0.874049	93.3739
3	0.940597	0.722353	0.711047	0.722353	0.71652	0.870303	0.45196	0.477724	0.464485	0.870303	72.3175
18	0.940241	0.720592	0.709387	0.720592	0.714812	0.872835	0.448832	0.474389	0.461257	0.872835	94.9744
9	0.940137	0.719327	0.708781	0.719327	0.713896	0.863974	0.447761	0.471823	0.459477	0.863974	93.7041
13	0.939583	0.71738	0.706329	0.71738	0.711678	0.856531	0.443063	0.468314	0.455338	0.856531	93.4076
8	0.939272	0.715797	0.704871	0.715797	0.71016	0.866576	0.440319	0.465305	0.452467	0.866576	93.4799

Table 6.1. *Isolation forest training data evaluation metrics table.*

Iteration	Accuracy	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	Macro-ROC	Precision	Recall	F1	ROC	Time
16	0.969937	0.792662	0.500043	0.792662	0.492457	0.905661	0.000088	0.615385	0.000176	0.905661	253.7361
4	0.968476	0.772701	0.500038	0.772701	0.492071	0.906247	0.000079	0.576923	0.000157	0.906247	198.8027
14	0.968226	0.791806	0.500041	0.791806	0.492012	0.91488	0.000083	0.615385	0.000166	0.91488	256.5248
19	0.968024	0.791705	0.50004	0.791705	0.491959	0.907029	0.000083	0.615385	0.000165	0.907029	253.1945
1	0.968016	0.77247	0.500038	0.77247	0.491952	0.912436	0.000077	0.576923	0.000155	0.912436	211.0259
5	0.967912	0.791649	0.50004	0.791649	0.49193	0.904008	0.000082	0.615385	0.000165	0.904008	221.3265
10	0.967791	0.791589	0.50004	0.791589	0.491898	0.911635	0.000082	0.615385	0.000164	0.911635	257.2346
7	0.967748	0.791567	0.50004	0.791567	0.491887	0.915841	0.000082	0.615385	0.000164	0.915841	255.7002
11	0.967697	0.791542	0.50004	0.791542	0.491874	0.905725	0.000082	0.615385	0.000164	0.905725	250.2062
2	0.967673	0.753068	0.500035	0.753068	0.491857	0.913306	0.000072	0.538462	0.000143	0.913306	201.7576
15	0.967632	0.791509	0.50004	0.791509	0.491856	0.910965	0.000082	0.615385	0.000163	0.910965	252.4534
6	0.967486	0.752975	0.500035	0.752975	0.491808	0.907191	0.000071	0.538462	0.000142	0.907191	254.02
12	0.967365	0.791376	0.50004	0.791376	0.491787	0.907348	0.000081	0.615385	0.000162	0.907348	256.3384
17	0.967353	0.791369	0.50004	0.791369	0.491784	0.8967	0.000081	0.615385	0.000162	0.8967	265.3844
20	0.967294	0.772109	0.500037	0.772109	0.491763	0.897381	0.000076	0.576923	0.000152	0.897381	256.608
3	0.967267	0.752865	0.500034	0.752865	0.491751	0.903518	0.000071	0.538462	0.000141	0.903518	208.0658
9	0.96672	0.752592	0.500034	0.752592	0.491609	0.911368	0.000069	0.538462	0.000139	0.911368	255.3841
18	0.966661	0.752562	0.500034	0.752562	0.491593	0.913511	0.000069	0.538462	0.000139	0.913511	256.8337
8	0.966291	0.771608	0.500036	0.771608	0.491502	0.906655	0.000074	0.576923	0.000147	0.906655	261.235
13	0.966268	0.790827	0.500038	0.790827	0.491501	0.900982	0.000078	0.615385	0.000157	0.900982	256.0704

Table 6.2. *Isolation forest test data evaluation metrics table.*

Both tables described above are descendingly rearranged according to the Macro-F1 score. The reason behind this sort is to highlight which models have higher precision and recall in combination. The first row of the table shows the best model according to the selected criterion. The time in the tables is measured in seconds, table 6.1 time represents the training time per particular model and table 6.2 represents the evaluation time per particular model when evaluating the test set. Below are visual plots of various evaluation metrics for isolation forest to understand the results in a detailed manner.

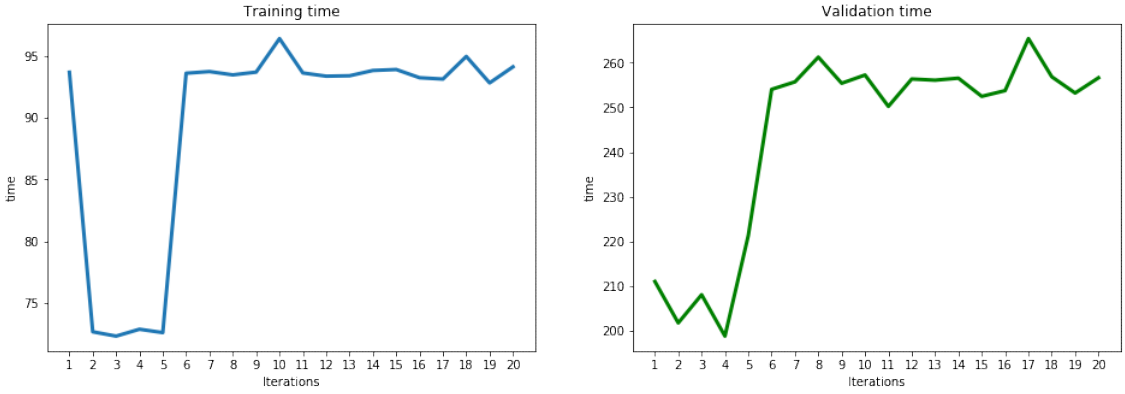

Figure 6.1. *Time elapsed for training and validating each model.*

Selecting a model based on these evaluation metrics is challenging; however, for this thesis, the Macro-F1 metric is selected as the baseline as we are interested in the optimized precision and recall combination of a model. By analyzing the visual illustrations and based on the highest macro-f1 metric and a fairly reasonable ROC curve and other evaluation metrics, iteration 16 is favourable.

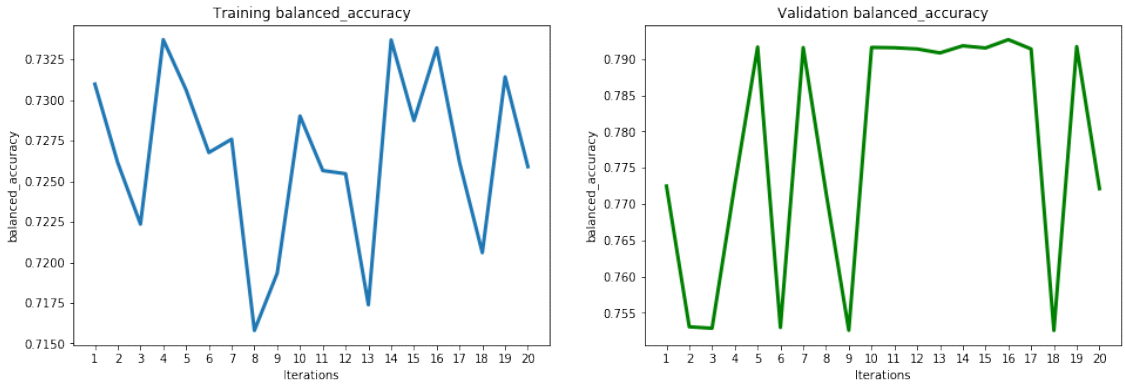

Figure 6.2. *Training and validation balanced accuracy metric for each model.*

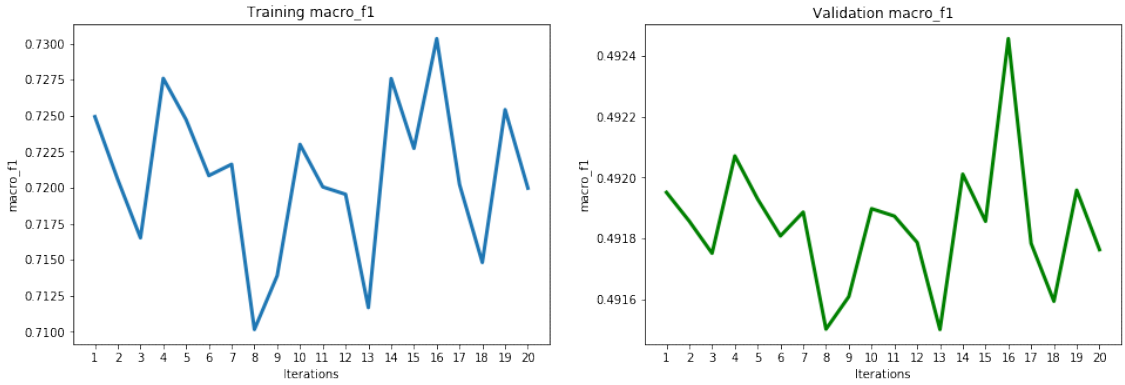

Figure 6.3. *Training and validation macro F1 score metric for each model.*

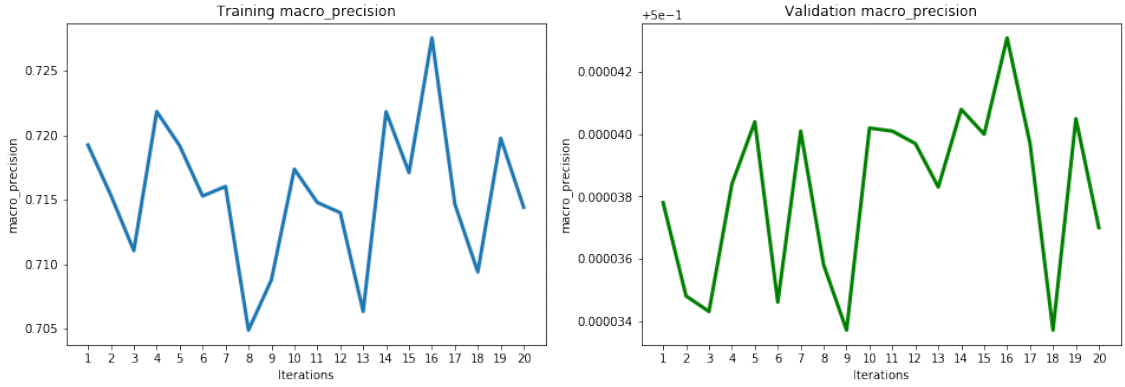

Figure 6.4. *Training and validation macro precision score metric for each model.*

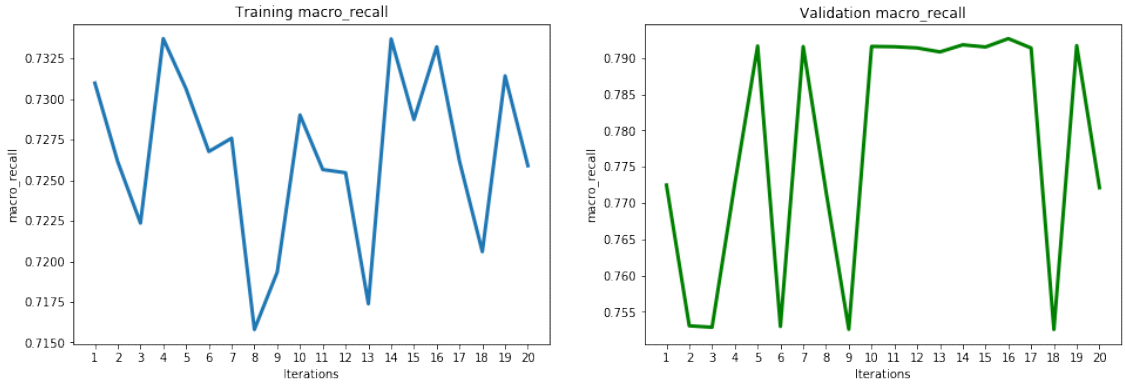

Figure 6.5. *Training and validation macro recall score metric for each model.*

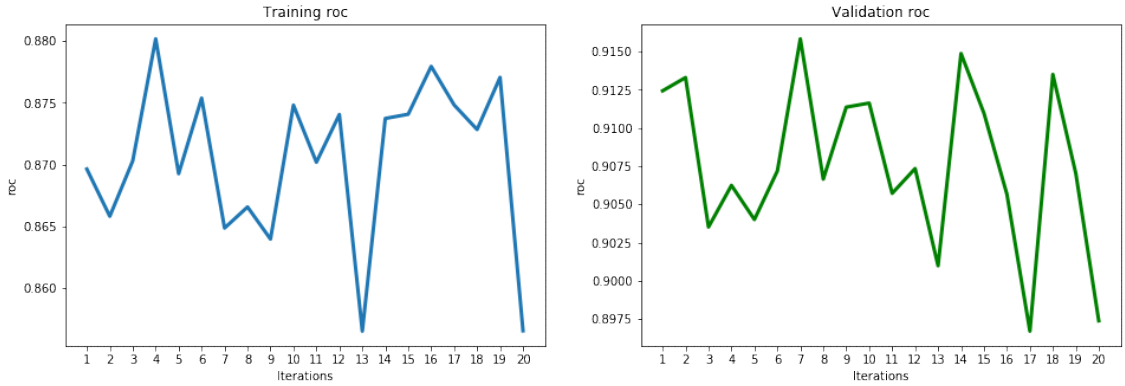

Figure 6.6. *Training and validation area under the curve (AUC) of receiver operating characteristic curve (ROC) for each model.*

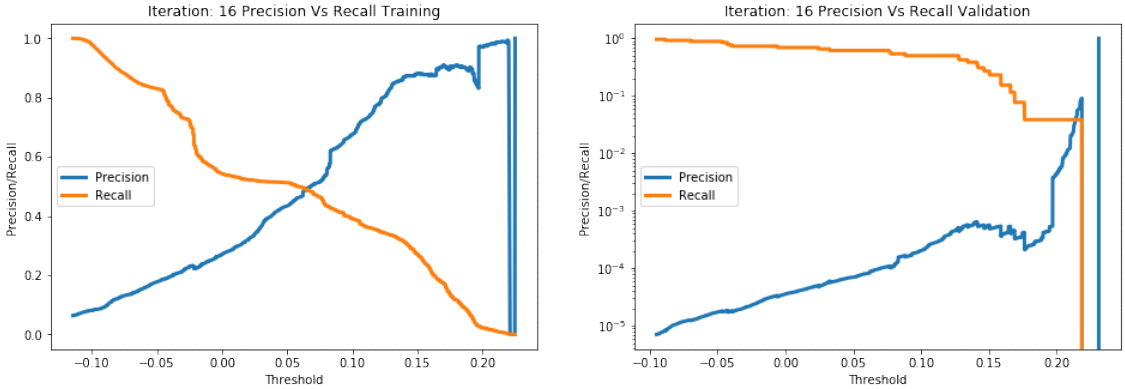

Figure 6.7. *Training and validation precision versus recall trade-off for iteration 16.*

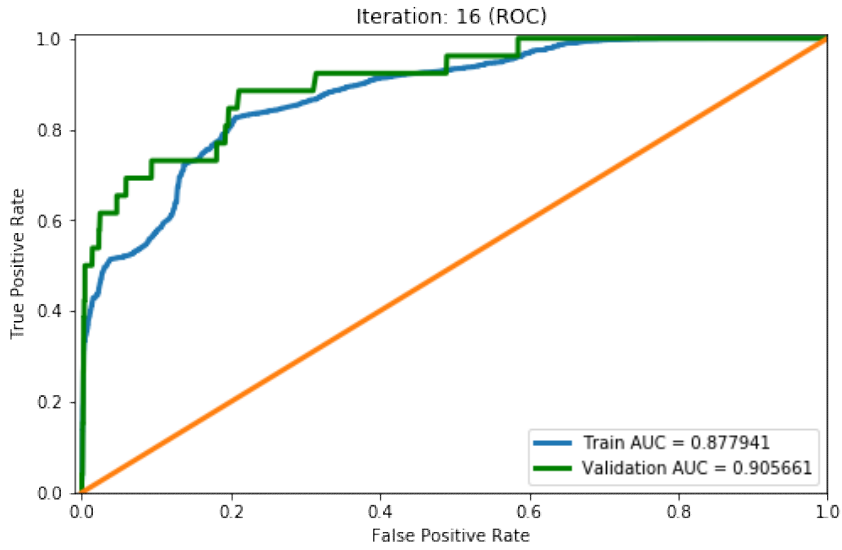

Figure 6.8. *Training and validation area under the curve (AUC) of receiver operating characteristic curve (ROC) for iteration 16.*

Confusion matrix gives an overall view of the classifer evaluation. Confusion metrics in figure 6.9 represent the classification evaluation in percentage format. Based on figure 6.9 it can be determined that 62% (16 out of 26) of the malicious transactions were correctly detected and equally important 97% (5,867,741 out of 6,049,601) of non-malicious transactions were also correctly classified as normal transactions.

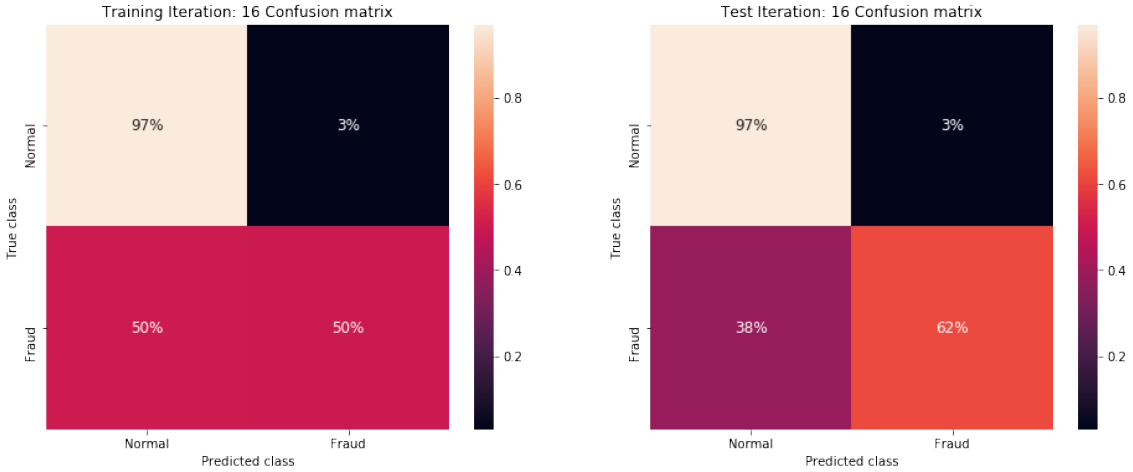

Figure 6.9. *Training and validation percentage confusion matrix for iteration 16.*

6.2 Histogram-based Outlier Score (HBOS)

As Goldstein and Dengel (2012) describe Histogram based outlier score (HBOS) is not so sensitive to the massive volume of data. The computation time for HBOS can be reasonable with large quantity of training data. However, to be fair about results comparisons with other algorithms, the training data is randomly subsampled into 20 chunks, and each chunk contains 1/10th of the data with substantial malicious data points. Chawla et al. (2002), Synthetic Minority Over-sampling Technique (SMOTE) is used to partially balance the class imbalance so the algorithm can search for a recognizable pattern. All hyper-parameters, including the over-sampling ratio, are estimated by the hyper-parameter tuning process. Best hyper-parameters are chosen based on the maximization of the macro-F1 metric. HBOS is estimated to give best results when trained with 8 N-bins, 0.714 alpha, 0.379 tol, and 0.249 contamination/oversampling ratio.

Iteration	Accuracy	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	Macro-ROC	Precision	Recall	F1	ROC	Time
15	0.85746	0.794962	0.77553	0.794962	0.784471	0.880219	0.629901	0.691021	0.659047	0.880219	1.1888
13	0.857261	0.794398	0.775246	0.794398	0.784068	0.880203	0.629603	0.689851	0.658351	0.880203	1.1697
12	0.856922	0.79378	0.774752	0.79378	0.78352	0.878854	0.628883	0.688767	0.657464	0.878854	1.1653
1	0.857022	0.793258	0.774919	0.793258	0.783397	0.879194	0.62954	0.687212	0.657113	0.879194	1.3421
5	0.85636	0.792043	0.773952	0.792043	0.782321	0.877926	0.62813	0.685076	0.655368	0.877926	1.1901
3	0.856291	0.792042	0.773846	0.792042	0.782259	0.877456	0.627903	0.685189	0.655296	0.877456	1.4542
14	0.847649	0.799336	0.762194	0.799336	0.777689	0.878736	0.59807	0.718987	0.652978	0.878736	1.1607
2	0.847627	0.798635	0.762111	0.798635	0.777398	0.877766	0.598315	0.717155	0.652367	0.877766	1.1706
4	0.845629	0.801872	0.759972	0.801872	0.77694	0.8776	0.591544	0.729101	0.653158	0.8776	1.3222
8	0.847228	0.798301	0.761586	0.798301	0.776929	0.877681	0.597354	0.716931	0.651702	0.877681	1.2399
6	0.84513	0.800333	0.75922	0.800333	0.77593	0.878612	0.590828	0.72583	0.651408	0.878612	1.1679
16	0.84524	0.7996	0.759276	0.7996	0.775741	0.878875	0.591409	0.723696	0.650899	0.878875	1.1703
20	0.844928	0.799925	0.758937	0.799925	0.775602	0.878035	0.590449	0.725081	0.650876	0.878035	1.1645
7	0.844992	0.799756	0.758996	0.799756	0.775592	0.877757	0.590688	0.724524	0.650796	0.877757	1.1771
9	0.84463	0.799311	0.758519	0.799311	0.775114	0.87831	0.589899	0.72394	0.650082	0.87831	1.2939
11	0.844504	0.799141	0.758352	0.799141	0.774943	0.875388	0.589632	0.723696	0.649821	0.875388	1.1674
17	0.843229	0.800405	0.757	0.800405	0.77435	0.878977	0.58581	0.729186	0.649682	0.878977	1.161
10	0.843886	0.79839	0.757539	0.79839	0.774131	0.874236	0.588284	0.722726	0.648612	0.874236	1.1543
19	0.842464	0.798853	0.755933	0.798853	0.773114	0.875649	0.584397	0.726325	0.647677	0.875649	1.3274
18	0.842004	0.798031	0.755303	0.798031	0.772412	0.874614	0.583505	0.724899	0.646562	0.874614	1.1794

Table 6.3. *Histogram based outlier score (HBOS) training data evaluation metrics table.*

Iteration	Accuracy	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	Macro-ROC	Precision	Recall	F1	ROC	Time
5	0.899034	0.814902	0.500015	0.814902	0.473448	0.913777	0.000031	0.730769	0.000062	0.913777	2.7831
3	0.899026	0.814898	0.500015	0.814898	0.473445	0.913763	0.000031	0.730769	0.000062	0.913763	2.4839
1	0.899022	0.814896	0.500015	0.814896	0.473444	0.913857	0.000031	0.730769	0.000062	0.913857	4.4509
15	0.898753	0.814762	0.500015	0.814762	0.47337	0.91402	0.000031	0.730769	0.000062	0.91402	2.6611
12	0.898675	0.814723	0.500015	0.814723	0.473348	0.913601	0.000031	0.730769	0.000062	0.913601	2.5596
13	0.89866	0.814715	0.500015	0.814715	0.473343	0.91819	0.000031	0.730769	0.000062	0.91819	2.564
2	0.880192	0.843942	0.500014	0.843942	0.468168	0.912984	0.000029	0.807692	0.000058	0.912984	2.5366
8	0.879901	0.843797	0.500014	0.843797	0.468086	0.913079	0.000029	0.807692	0.000058	0.913079	2.5057
14	0.879889	0.843791	0.500014	0.843791	0.468083	0.913121	0.000029	0.807692	0.000058	0.913121	3.0518
16	0.87542	0.803095	0.500012	0.803095	0.466811	0.910772	0.000025	0.730769	0.00005	0.910772	2.5402
4	0.87469	0.80273	0.500012	0.80273	0.466603	0.912464	0.000025	0.730769	0.00005	0.912464	2.492
9	0.874591	0.80268	0.500012	0.80268	0.466575	0.910173	0.000025	0.730769	0.00005	0.910173	2.4895
7	0.874567	0.802668	0.500012	0.802668	0.466568	0.910708	0.000025	0.730769	0.00005	0.910708	2.5446
20	0.874507	0.802638	0.500012	0.802638	0.466551	0.910553	0.000025	0.730769	0.00005	0.910553	2.533
6	0.874503	0.802636	0.500012	0.802636	0.46655	0.910531	0.000025	0.730769	0.00005	0.910531	2.5219
11	0.874265	0.802518	0.500012	0.802518	0.466483	0.910078	0.000025	0.730769	0.00005	0.910078	2.5244
10	0.87421	0.80249	0.500012	0.80249	0.466467	0.910108	0.000025	0.730769	0.00005	0.910108	2.4964
18	0.871317	0.839505	0.500013	0.839505	0.465644	0.912691	0.000027	0.807692	0.000054	0.912691	2.513
17	0.871294	0.839493	0.500013	0.839493	0.465637	0.913317	0.000027	0.807692	0.000054	0.913317	2.5208
19	0.871152	0.839422	0.500013	0.839422	0.465597	0.912713	0.000027	0.807692	0.000054	0.912713	2.5259

Table 6.4. *Histogram based outlier score (HBOS) test data evaluation metrics table.*

Table 6.3 describes the metrics obtained from evaluating the training data for each model, whereas the table 6.4 describes evaluation results from the test set for each model. Both tables are sorted in a descending manner based on the macro-F1 score, and time complexity is measured in seconds. Based on training evaluation metrics, iteration 15 performs better, but on the other hand based on test evaluation metrics, iteration 5 seems to have more noticeable results. Upon more in-depth analysis, it is observed that the gap between both the evaluation model is minor as compared to other models. The trade-off of macro-f1 score and area under the curve (AUC) for receiver operating characteristic (ROC) has to be balanced along with other metrics while choosing an optimal model. Therefore based on the plots iteration 17 is chosen as the baseline model. Training time and evaluation time for the histogram-based outlier score (HBOS) is comparatively shorter for the isolation forest algorithm. However, increasing the training set size for each iteration has not caused a significant impact on the overall performance of the models.

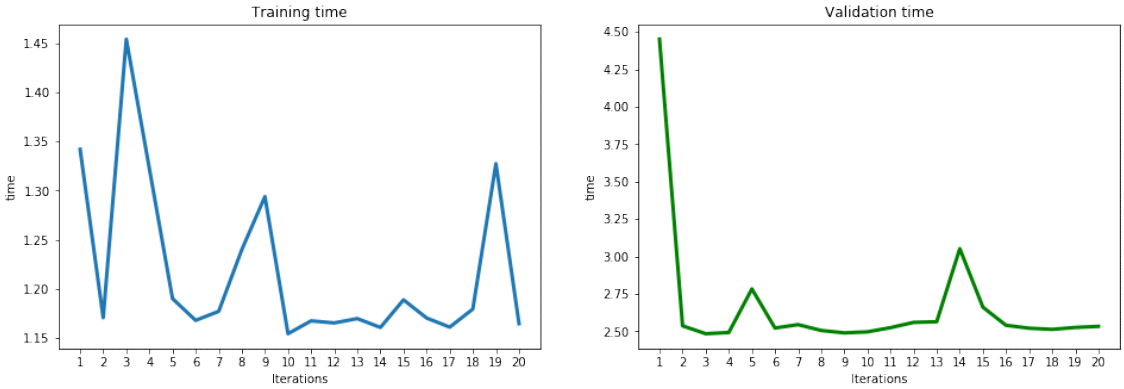

Figure 6.10. *Time elapsed for training and validating each model.*

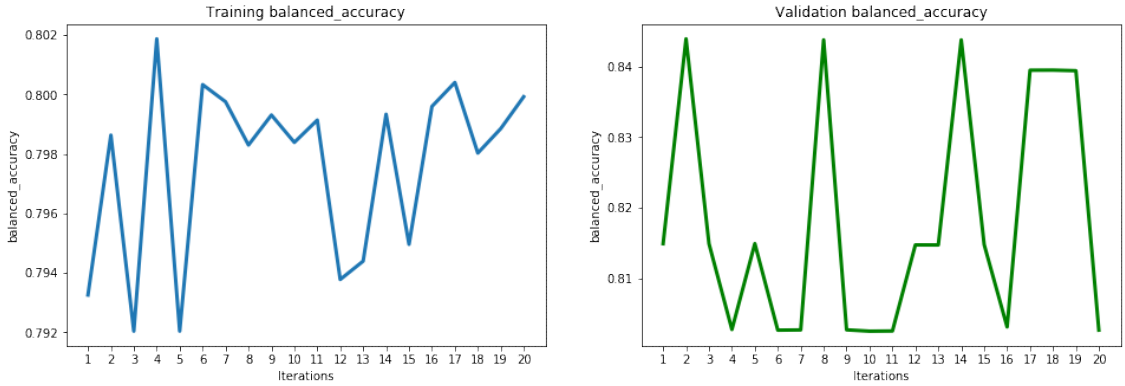

Figure 6.11. *Training and validation balanced accuracy metric for each model.*

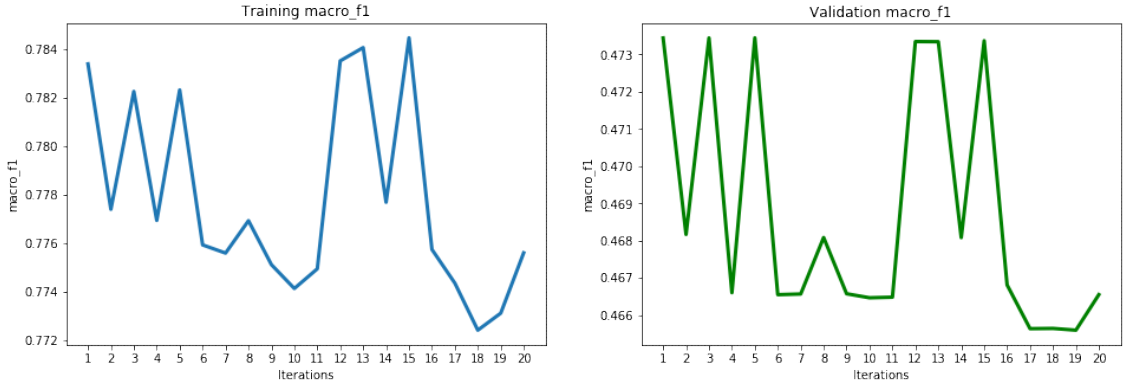

Figure 6.12. *Training and validation macro F1 score metric for each model.*

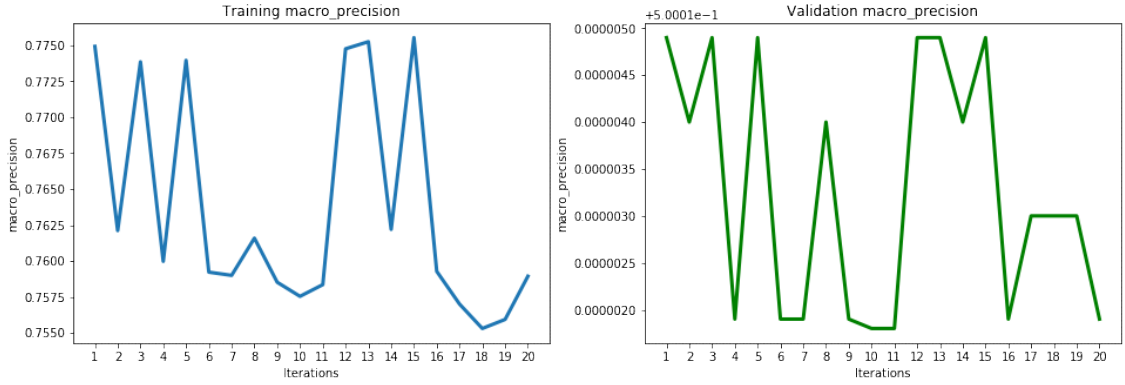

Figure 6.13. *Training and validation macro precision score metric for each model.*

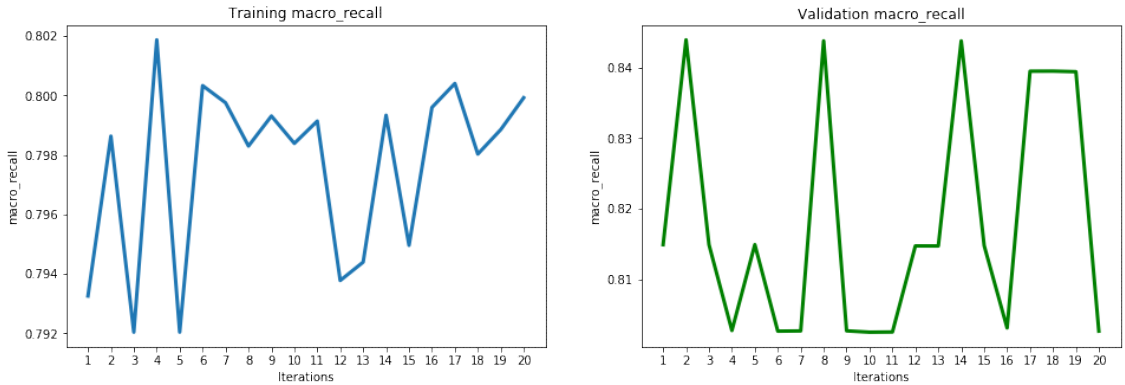

Figure 6.14. *Training and validation macro recall score metric for each model.*

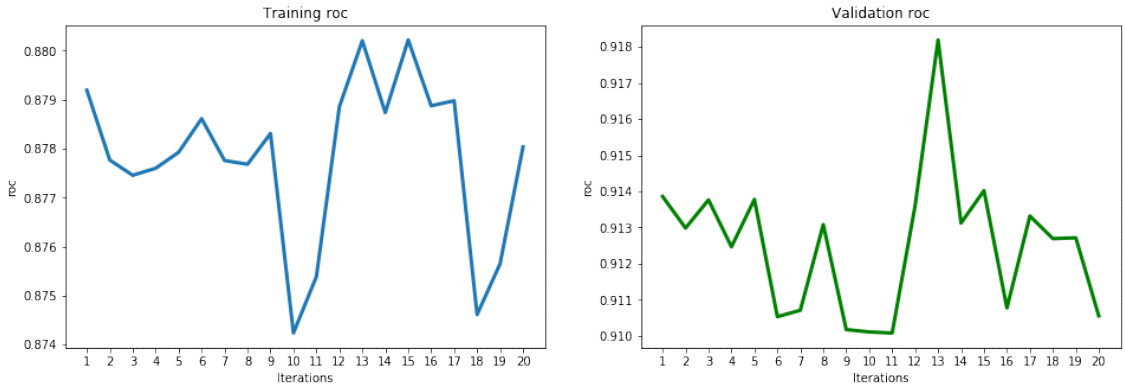

Figure 6.15. *Training and validation area under the curve (AUC) of receiver operating characteristic curve (ROC) for each model.*

Visual illustrations below are a deeper dive into the selected model, which is of iteration 17. The validation receiver operating characteristic (ROC) has performed significantly better than training receiver operating characteristic (ROC) for histogram-based outlier score (HBOS). Finding the optimal threshold for the model is a challenging task and requires extensive research and experimentation of its own. Nevertheless, this library used by this thesis to detect anomalies has a built-in algorithm, which optimally selects the threshold with the best possible results. Figure 6.16 shows interesting behaviour near high threshold values, the model initially starts going towards underfitting and then after that toward overfitting. This happens due the increasing value of threshold with respect to the model evaluation.

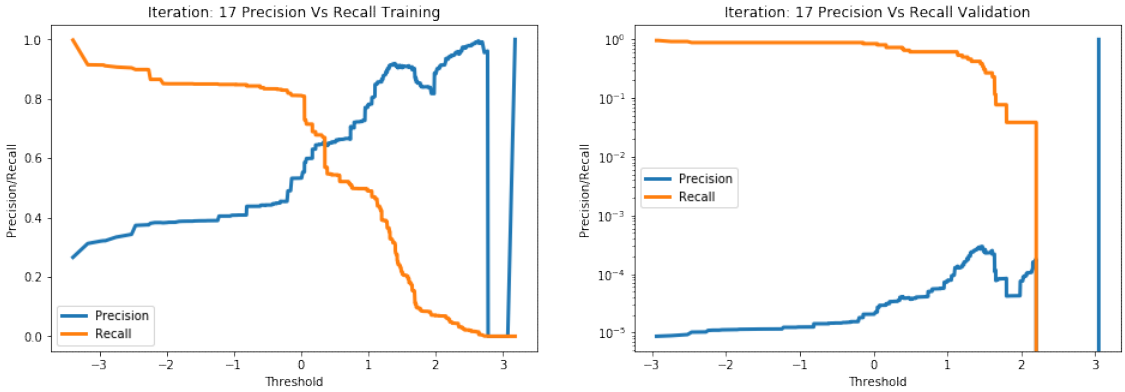

Figure 6.16. *Training and validation precision versus recall trade-off for iteration 17.*

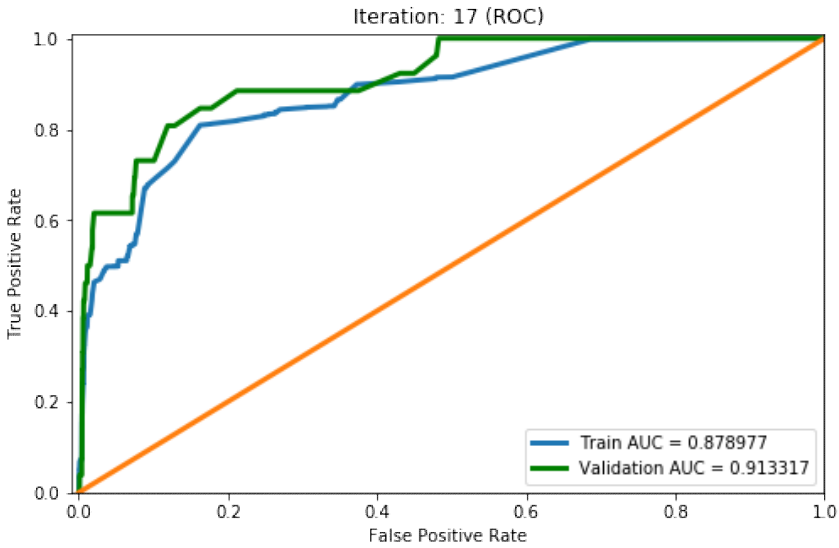

Figure 6.17. *Training and validation area under the curve (AUC) of receiver operating characteristic curve (ROC) for iteration 17.*

Confusion matrix for histogram-based outlier score (HBOS) revealed that although 81% (21 out of 26) of malicious transactions were detected successfully and 87% (5,270,982 out of 6,049,601) of non-malicious transactions were also detected correctly in the test set, also the false-positive rate was a fairly moderate which is 19% (5 out of 26) for this algorithm. For both training and test evaluation, the model seems to perform with moderate false positives and false negatives. Although the significance of performance metrics is wholly based on the use-case, this thesis in an ideal use-case ultimately aims to maximize the detection of true positives and true negatives while minimizing the false positives.

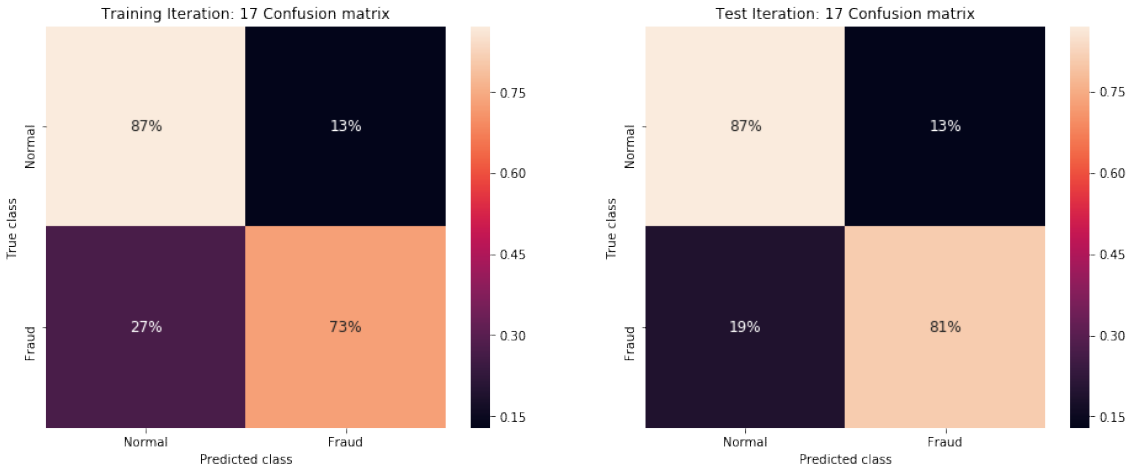

Figure 6.18. *Training and validation percentage confusion matrix for iteration 17.*

6.3 Cluster-Based Local Outlier Factor (CBLOF)

He, Xu, and Deng (2003) describe Cluster-based local outlier factor (CBLOF) is sensitive to the massive data volumes. The computation time for CBLOF can be enormous with a large quantity of training data. However, to overcome that, the training data is randomly subsampled into 20 subsets and each subset contains 1/10th of the data.

Iteration	Accuracy	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	Macro-ROC	Precision	Recall	F1	ROC	Time
3	0.800226	0.721385	0.689702	0.721385	0.702203	0.758768	0.480692	0.593944	0.531351	0.758768	47.9858
15	0.79995	0.720938	0.689319	0.720938	0.701793	0.758496	0.480108	0.593221	0.530704	0.758496	45.6294
12	0.799946	0.720931	0.689313	0.720931	0.701786	0.75838	0.480098	0.593209	0.530693	0.75838	47.363
6	0.799931	0.720907	0.689292	0.720907	0.701764	0.757914	0.480066	0.593171	0.530659	0.757914	46.7367
7	0.79982	0.720727	0.689138	0.720727	0.701598	0.758771	0.479831	0.592879	0.530398	0.758771	41.7685
19	0.799788	0.720676	0.689094	0.720676	0.701551	0.758266	0.479764	0.592797	0.530324	0.758266	46.5725
17	0.79978	0.720662	0.689082	0.720662	0.701538	0.758066	0.479745	0.592774	0.530304	0.758066	49.5699
1	0.799772	0.720649	0.689071	0.720649	0.701527	0.757941	0.479729	0.592753	0.530285	0.757941	49.8873
13	0.79976	0.72063	0.689055	0.72063	0.701508	0.758434	0.479703	0.592722	0.530257	0.758434	43.0998
4	0.799732	0.720584	0.689016	0.720584	0.701467	0.758568	0.479643	0.592648	0.530191	0.758568	39.1775
11	0.799678	0.720497	0.688941	0.720497	0.701387	0.758337	0.47953	0.592507	0.530065	0.758337	44.2574
9	0.799612	0.720389	0.688849	0.720389	0.701288	0.75799	0.479389	0.592332	0.529909	0.75799	38.8703
14	0.799536	0.720268	0.688744	0.720268	0.701175	0.757593	0.479229	0.592136	0.529733	0.757593	39.4134
16	0.799524	0.720248	0.688728	0.720248	0.701157	0.757786	0.479203	0.592104	0.529705	0.757786	42.8584
20	0.799517	0.720236	0.688717	0.720236	0.701147	0.75754	0.479188	0.592085	0.529687	0.75754	41.345
5	0.799494	0.720199	0.688686	0.720199	0.701113	0.757807	0.479139	0.592025	0.529634	0.757807	41.228
18	0.799475	0.720166	0.688659	0.720166	0.701083	0.757947	0.479099	0.591969	0.529587	0.757947	39.2051
10	0.799406	0.720057	0.688564	0.720057	0.700982	0.757733	0.478953	0.591795	0.529428	0.757733	41.2255
2	0.797079	0.716287	0.685334	0.716287	0.697513	0.760188	0.474015	0.585693	0.523969	0.760188	46.7029
8	0.795934	0.714432	0.683744	0.714432	0.695806	0.763466	0.471584	0.58269	0.521283	0.763466	47.5644

Table 6.5. *Cluster-based local outlier factor (CBLOF) training data evaluation metrics table.*

Iteration	Accuracy	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	Macro-ROC	Precision	Recall	F1	ROC	Time
7	0.848902	0.789836	0.50001	0.789836	0.459159	0.818955	0.000021	0.730769	0.000042	0.818955	4.8076
13	0.848684	0.789727	0.50001	0.789727	0.459095	0.81956	0.000021	0.730769	0.000041	0.81956	4.8853
6	0.848673	0.789721	0.50001	0.789721	0.459092	0.81903	0.000021	0.730769	0.000041	0.81903	5.0782
4	0.848649	0.789709	0.50001	0.789709	0.459085	0.818928	0.000021	0.730769	0.000041	0.818928	4.835
17	0.848617	0.789694	0.50001	0.789694	0.459076	0.819533	0.000021	0.730769	0.000041	0.819533	4.8359
12	0.848572	0.789671	0.50001	0.789671	0.459063	0.819083	0.000021	0.730769	0.000041	0.819083	4.7862
9	0.848565	0.789667	0.50001	0.789667	0.45906	0.819022	0.000021	0.730769	0.000041	0.819022	4.9408
5	0.848348	0.789559	0.50001	0.789559	0.458997	0.81927	0.000021	0.730769	0.000041	0.81927	4.8933
1	0.848344	0.789557	0.50001	0.789557	0.458996	0.819057	0.000021	0.730769	0.000041	0.819057	5.1392
15	0.848343	0.789556	0.50001	0.789556	0.458995	0.819055	0.000021	0.730769	0.000041	0.819055	4.814
19	0.848323	0.789547	0.50001	0.789547	0.45899	0.819824	0.000021	0.730769	0.000041	0.819824	5.2029
14	0.848322	0.789546	0.50001	0.789546	0.458989	0.819066	0.000021	0.730769	0.000041	0.819066	4.9586
11	0.848317	0.789544	0.50001	0.789544	0.458988	0.819092	0.000021	0.730769	0.000041	0.819092	5.2527
10	0.848304	0.789537	0.50001	0.789537	0.458984	0.819079	0.000021	0.730769	0.000041	0.819079	4.9892
3	0.848294	0.789532	0.50001	0.789532	0.458981	0.819352	0.000021	0.730769	0.000041	0.819352	5.0132
18	0.848213	0.789491	0.50001	0.789491	0.458957	0.818937	0.000021	0.730769	0.000041	0.818937	4.8327
20	0.848201	0.789485	0.50001	0.789485	0.458954	0.819166	0.000021	0.730769	0.000041	0.819166	4.937
16	0.848077	0.789423	0.50001	0.789423	0.458918	0.819653	0.000021	0.730769	0.000041	0.819653	4.9704
2	0.8468	0.788785	0.50001	0.788785	0.458543	0.824439	0.000021	0.730769	0.000041	0.824439	4.9829
8	0.845836	0.788303	0.500009	0.788303	0.45826	0.848228	0.00002	0.730769	0.000041	0.848228	4.9508

Table 6.6. *Cluster-based local outlier factor (CBLOF) test data evaluation metrics table.*

Each training subset is oversampled with substantial malicious data points by using the Synthetic Minority Over-sampling Technique (SMOTE). The estimated optimal hyper-parameters computed for CBLOF are 8 N-clusters, 0.2356 contamination, and outliers fraction with 0.839 alpha and 2 beta. Based on descriptive

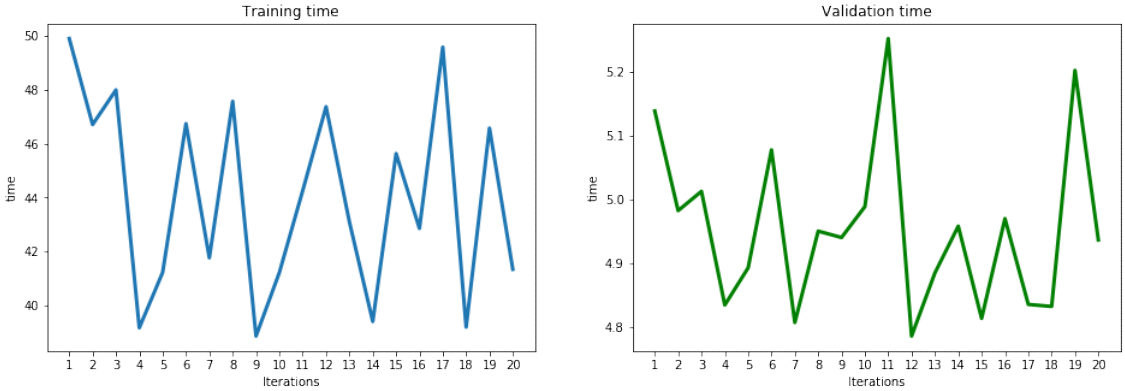

Figure 6.19. *Time elapsed for training and validating each model.*

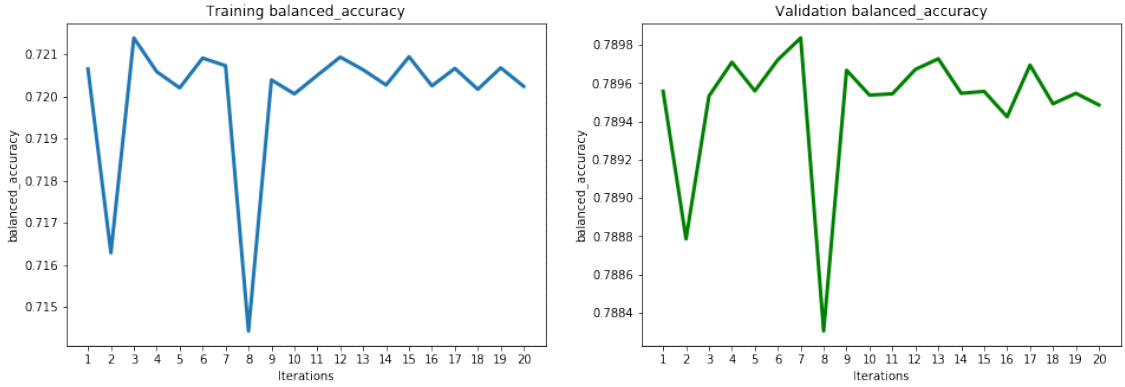

Figure 6.20. *Training and validation balanced accuracy metric for each model.*

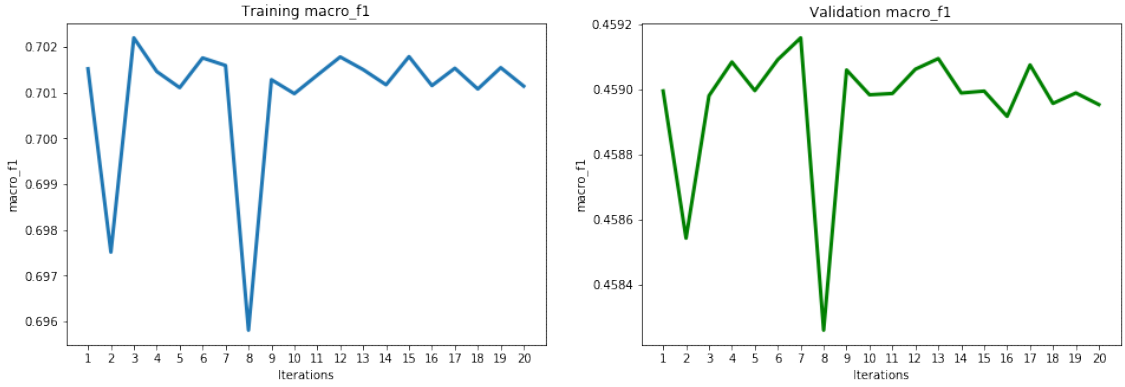

Figure 6.21. *Training and validation macro F1 score metric for each model.*

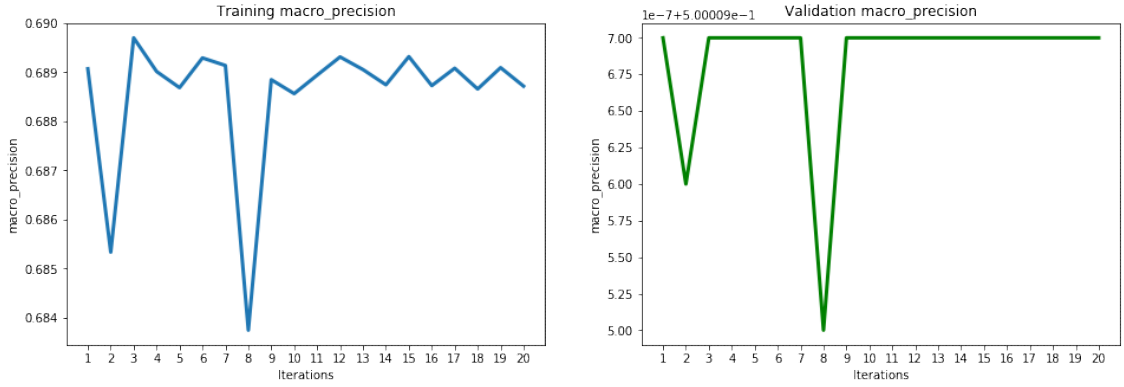

Figure 6.22. *Training and validation macro precision score metric for each model.*

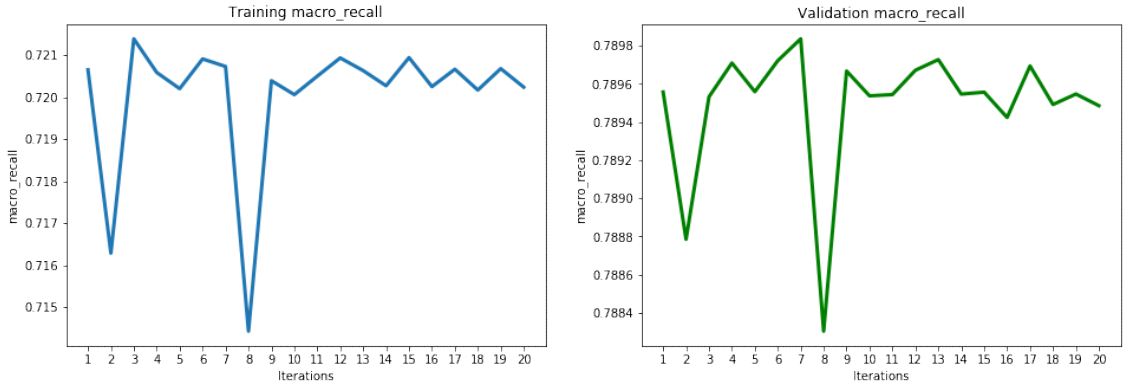

Figure 6.23. *Training and validation macro recall score metric for each model.*

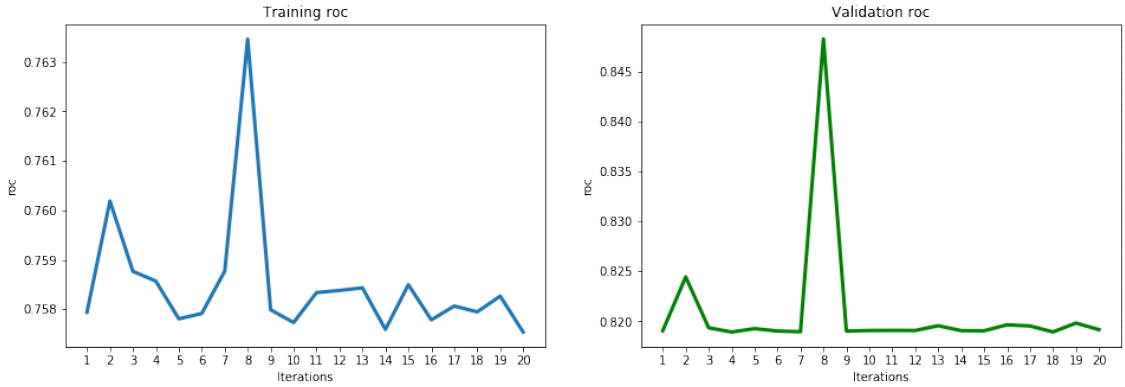

Figure 6.24. *Training and validation area under the curve (AUC) of receiver operating characteristic curve (ROC) for each model.*

tables and evaluation visualizations above, iteration 7 is selected as the CBLOF base model.

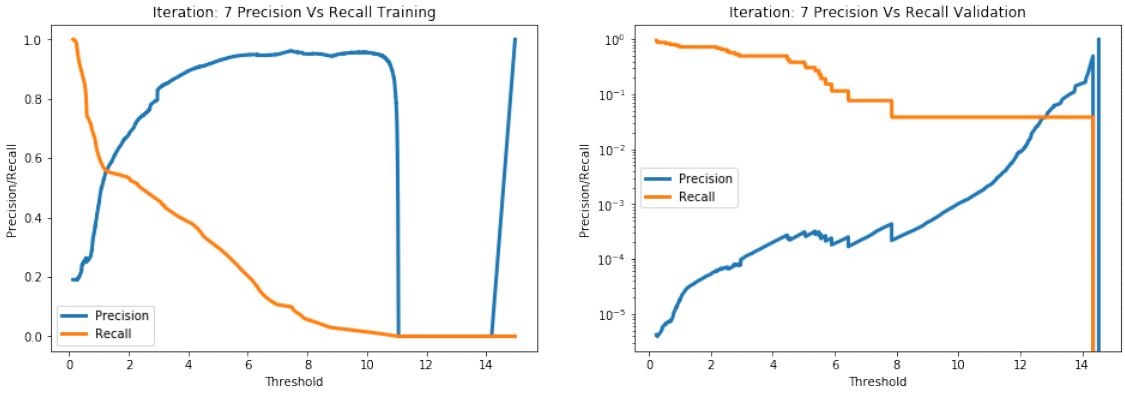

Figure 6.25. *Training and validation precision versus recall trade-off for iteration 7.*

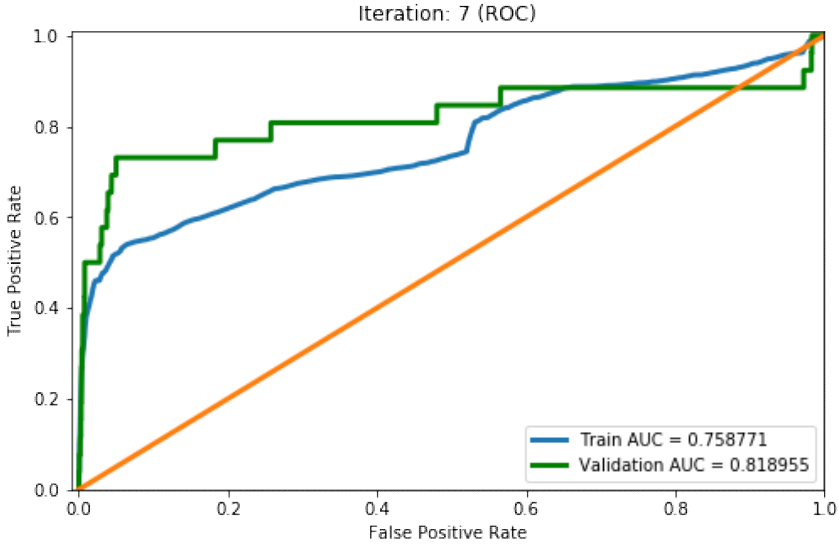

Figure 6.26. *Training and validation area under the curve (AUC) of receiver operating characteristic curve (ROC) for iteration 7.*

The confusion matrix below for cluster-based local outlier factor (CBLOF) shows that the algorithm performs well. The overall results with 73% (19 out of 26) successful detection of malicious data points along with 85% (5,135,522 out of 6,049,601) of successful detection of non-malicious data points are satisfactory. Only 27% (7 out of 26) false-positive cases were identified in the test set which is also adequate for this use-case.

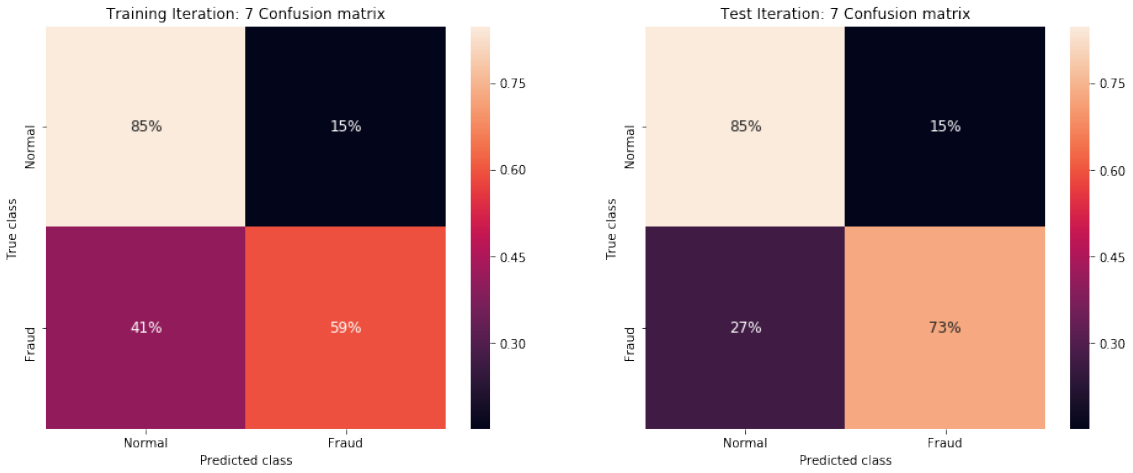

Figure 6.27. *Training and validation percentage confusion matrix for iteration 7.*

6.4 Principal Component Analysis (PCA)

Jolliffe (2011) explain Principal component analysis (PCA) is sensitive to the massive data volumes. But for our training data the computation time for PCA is reasonable. However, to have a fair evaluation comparison, training data is randomly subsampled into 20 subsets where each subset is the size of 1/10th of the total data. Table 6.7 and table 6.8 below describe the evaluation metrics for training and test data. Each training subset was partially balanced with malicious data points by using the Synthetic Minority Over-sampling Technique (SMOTE). The estimated optimal hyper-parameters computed for PCA are 3 N-components with outliers-fraction of 0.240 and tol value of 0.5. Based on descriptive tables and evaluation visualizations below, iteration 19 is selected as the PCA base model.

Iteration	Accuracy	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	Macro-ROC	Precision	Recall	F1	ROC	Time
19	0.855803	0.814644	0.769254	0.814644	0.787675	0.893438	0.602815	0.747491	0.667403	0.893438	5.5105
5	0.855719	0.814511	0.76914	0.814511	0.787552	0.8931	0.602641	0.747276	0.66721	0.8931	5.7261
10	0.855663	0.81442	0.769063	0.81442	0.787469	0.893047	0.602523	0.74713	0.66708	0.893047	5.5939
17	0.855622	0.814354	0.769006	0.814354	0.787408	0.892859	0.602437	0.747023	0.666984	0.892859	5.2143
4	0.855602	0.814323	0.768979	0.814323	0.787379	0.893145	0.602397	0.746973	0.66694	0.893145	5.8239
7	0.855552	0.814243	0.768911	0.814243	0.787306	0.892894	0.602293	0.746844	0.666825	0.892894	5.1902
1	0.855533	0.814212	0.768884	0.814212	0.787277	0.893143	0.602253	0.746794	0.66678	0.893143	6.6694
15	0.855521	0.814193	0.768868	0.814193	0.787259	0.893068	0.602228	0.746763	0.666752	0.893068	5.4579
11	0.855434	0.814053	0.768748	0.814053	0.787131	0.892791	0.602046	0.746537	0.666551	0.892791	5.5301
8	0.855268	0.813787	0.768521	0.813787	0.786887	0.892802	0.6017	0.746109	0.666168	0.892802	6.4253
6	0.851816	0.808259	0.76379	0.808259	0.781805	0.896883	0.59451	0.737193	0.658207	0.896883	5.4997
14	0.851678	0.808037	0.7636	0.808037	0.7816	0.896849	0.594221	0.736835	0.657888	0.896849	5.3444
13	0.85166	0.80801	0.763577	0.80801	0.781575	0.8967	0.594185	0.73679	0.657848	0.8967	5.6395
18	0.851602	0.807904	0.763496	0.807904	0.781485	0.896612	0.594069	0.736607	0.657704	0.896612	5.4828
20	0.851484	0.807727	0.763335	0.807727	0.781315	0.896566	0.593817	0.736333	0.65744	0.896566	5.5712
16	0.851342	0.807499	0.76314	0.807499	0.781106	0.896338	0.593521	0.735967	0.657113	0.896338	5.7013
3	0.851282	0.807404	0.763059	0.807404	0.781018	0.896527	0.593398	0.735814	0.656976	0.896527	5.3711
2	0.851282	0.807403	0.763058	0.807403	0.781018	0.896273	0.593396	0.735812	0.656975	0.896273	6.0793
9	0.851241	0.807338	0.763002	0.807338	0.780958	0.896373	0.593311	0.735707	0.656881	0.896373	5.4573
12	0.851177	0.807236	0.762914	0.807236	0.780863	0.896528	0.593178	0.735541	0.656733	0.896528	5.3046

Table 6.7. *Principal component analysis (PCA) training data evaluation metrics table.*

Iteration	Accuracy	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	Macro-ROC	Precision	Recall	F1	ROC	Time
8	0.88197	0.883293	0.500016	0.883293	0.468674	0.909814	0.000032	0.884615	0.000064	0.909814	0.9322
1	0.881901	0.883258	0.500016	0.883258	0.468654	0.910061	0.000032	0.884615	0.000064	0.910061	1.1839
19	0.881852	0.883234	0.500016	0.883234	0.468641	0.910491	0.000032	0.884615	0.000064	0.910491	0.8296
10	0.881808	0.883212	0.500016	0.883212	0.468628	0.909745	0.000032	0.884615	0.000064	0.909745	0.9687
17	0.881808	0.883212	0.500016	0.883212	0.468628	0.909967	0.000032	0.884615	0.000064	0.909967	0.843
4	0.881792	0.883204	0.500016	0.883204	0.468624	0.910157	0.000032	0.884615	0.000064	0.910157	0.8573
5	0.881757	0.883186	0.500016	0.883186	0.468614	0.909917	0.000032	0.884615	0.000064	0.909917	0.883
11	0.881737	0.883176	0.500016	0.883176	0.468608	0.910082	0.000032	0.884615	0.000064	0.910082	0.9452
7	0.881679	0.883147	0.500016	0.883147	0.468592	0.909964	0.000032	0.884615	0.000064	0.909964	0.8656
15	0.881676	0.883146	0.500016	0.883146	0.468591	0.910166	0.000032	0.884615	0.000064	0.910166	0.8763
14	0.879623	0.862889	0.500015	0.862889	0.468009	0.917315	0.00003	0.846154	0.00006	0.917315	0.8991
13	0.879504	0.862829	0.500015	0.862829	0.467975	0.91736	0.00003	0.846154	0.00006	0.91736	0.8642
18	0.879397	0.862776	0.500015	0.862776	0.467944	0.917235	0.00003	0.846154	0.00006	0.917235	0.8393
6	0.879365	0.862759	0.500015	0.862759	0.467935	0.917357	0.00003	0.846154	0.00006	0.917357	0.9632
16	0.879336	0.862745	0.500015	0.862745	0.467927	0.91722	0.00003	0.846154	0.00006	0.91722	1.1638
9	0.879199	0.862676	0.500015	0.862676	0.467888	0.9172	0.00003	0.846154	0.00006	0.9172	0.9278
3	0.879174	0.862664	0.500015	0.862664	0.467881	0.917208	0.00003	0.846154	0.00006	0.917208	0.8253
2	0.879044	0.862599	0.500015	0.862599	0.467844	0.917239	0.00003	0.846154	0.00006	0.917239	0.8565
12	0.878905	0.862529	0.500015	0.862529	0.467805	0.917177	0.00003	0.846154	0.00006	0.917177	0.8492
20	0.878859	0.862506	0.500015	0.862506	0.467792	0.91719	0.00003	0.846154	0.00006	0.91719	0.8788

Table 6.8. *Principal component analysis (PCA) test data evaluation metrics table.*

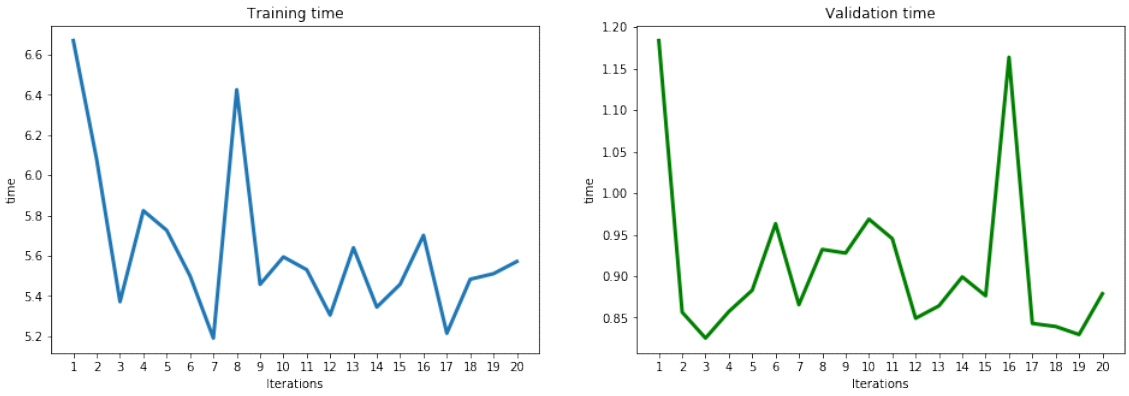

Figure 6.28. *Time elapsed for training and validating each model.*

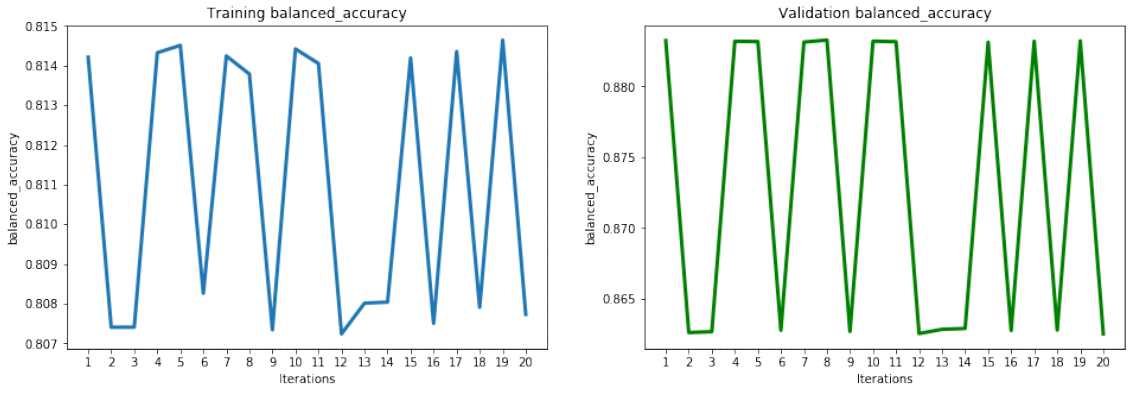

Figure 6.29. *Training and validation balanced accuracy metric for each model.*

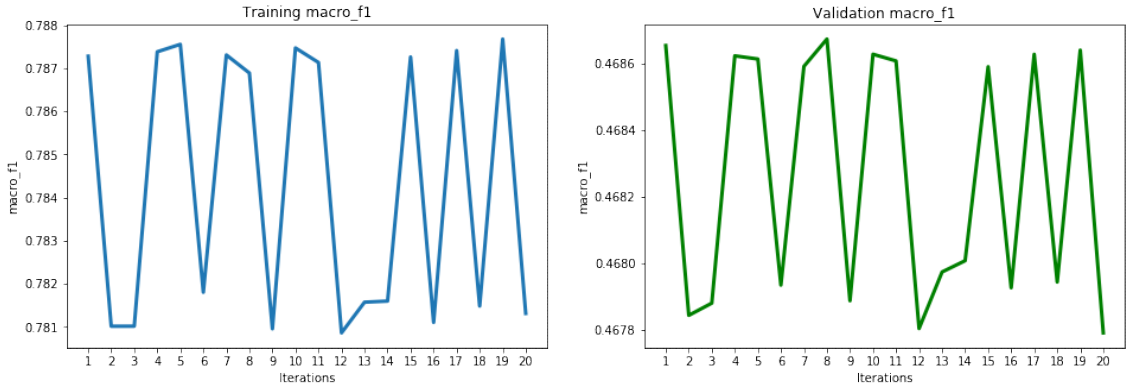

Figure 6.30. *Training and validation macro F1 score metric for each model.*

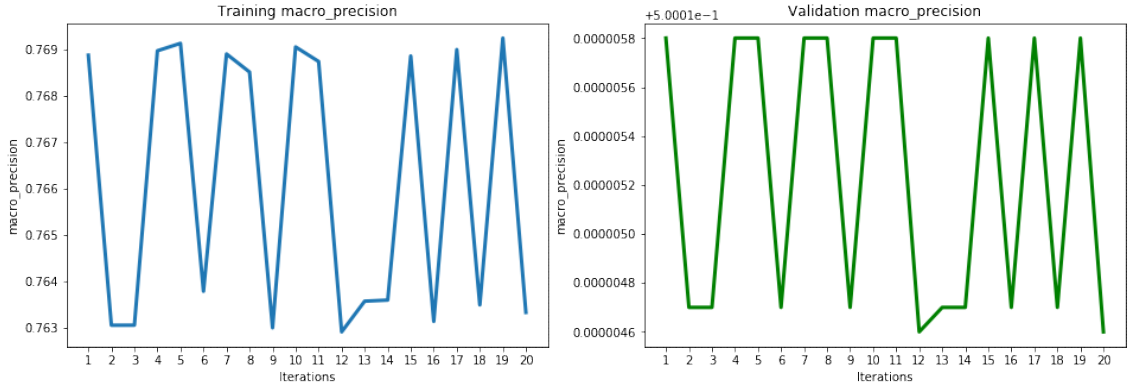

Figure 6.31. *Training and validation macro precision score metric for each model.*

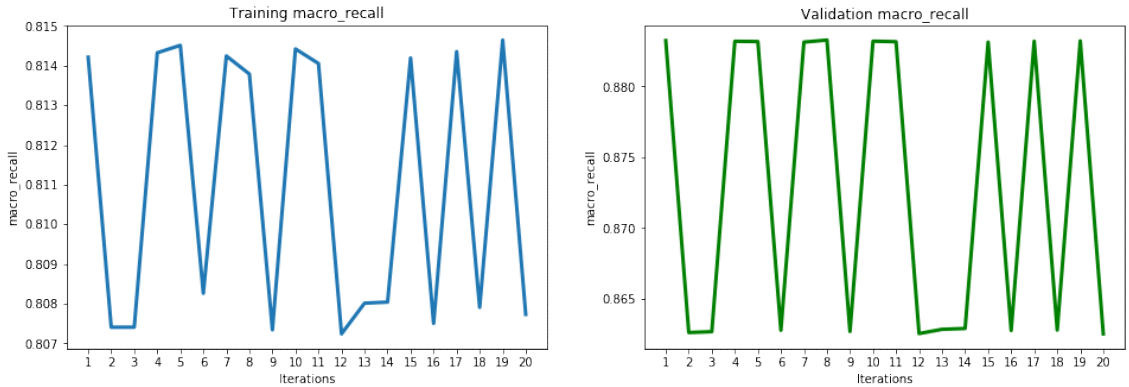

Figure 6.32. *Training and validation macro recall score metric for each model.*

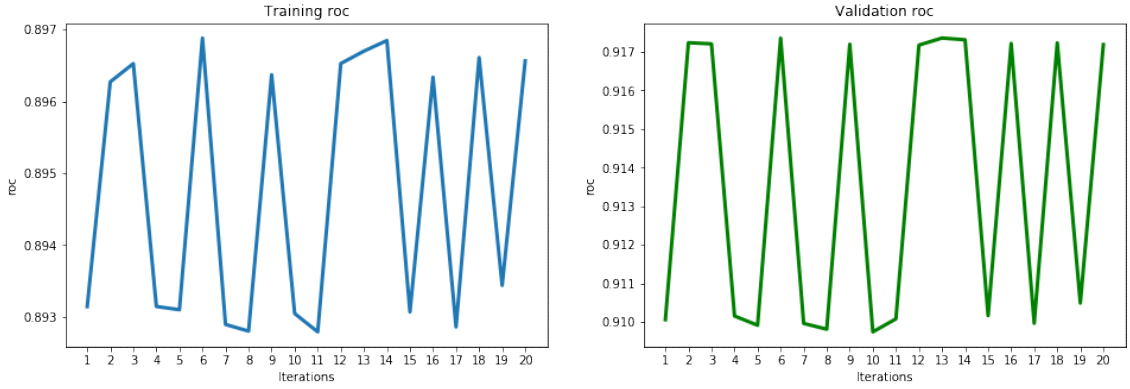

Figure 6.33. *Training and validation area under the curve (AUC) of receiver operating characteristic curve (ROC) for each model.*

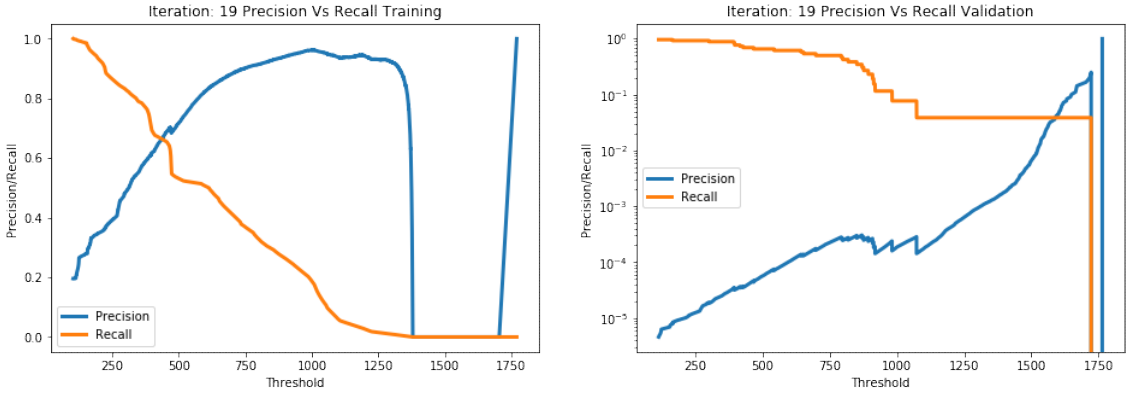

Figure 6.34. *Training and validation precision versus recall trade-off for iteration 19.*

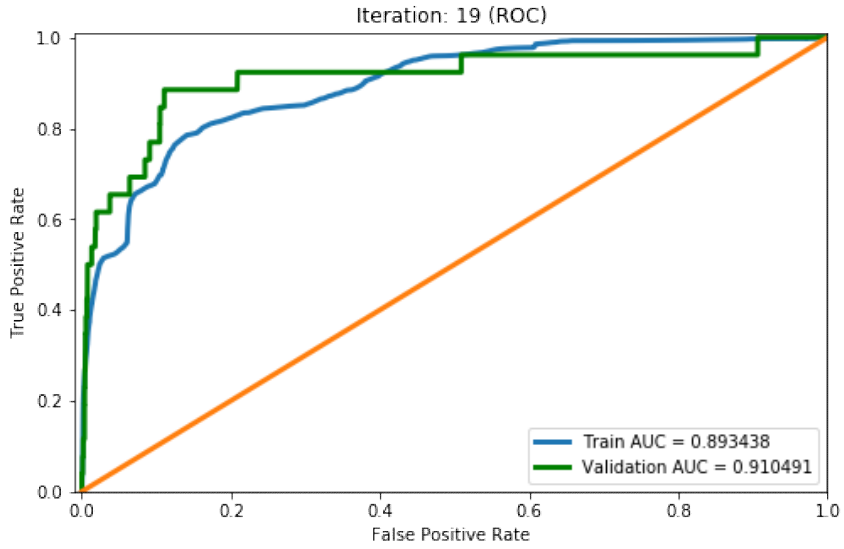

Figure 6.35. *Training and validation area under the curve (AUC) of receiver operating characteristic curve (ROC) for iteration 19.*

The confusion matrix below for principal component analysis (PCA) attests that the algorithm performs admirably. Evaluation of the training set revealed that the model was able to successfully identify 75% (434,116 out of 580,764) of malicious transactions with a considerable rate of 25% (146,648 out of 580,764) false-positives. However, 88% (Malicious: 23 out of 26 and Non-Malicious: 5,334,853 out of 6,049,601) of both malicious and non-malicious data points were successfully detected in test data, with a low percentage of 12% (Malicious: 3 out of 26 and 714,748 out of 6,049,601) of false-positive cases.

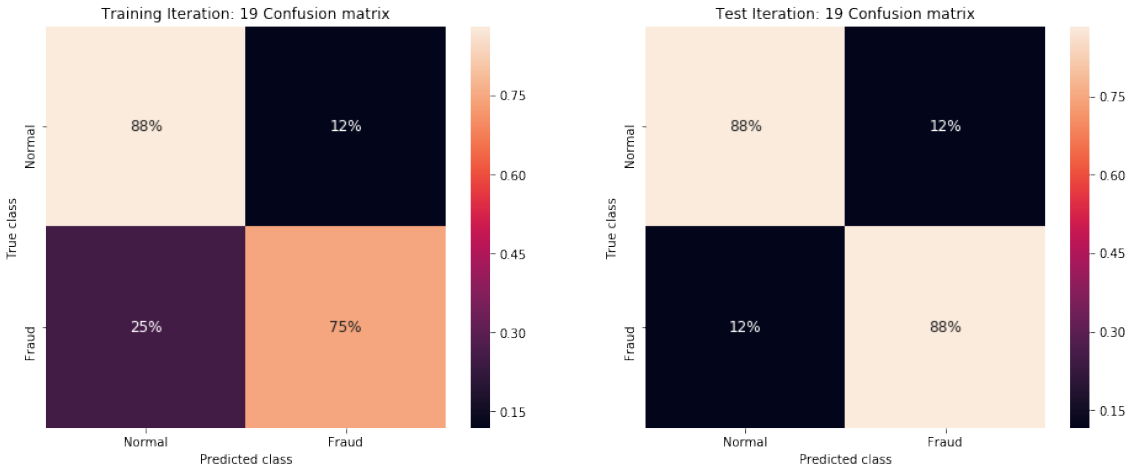

Figure 6.36. *Training and validation percentage confusion matrix for iteration 19.*

6.5 K-means

Arthur and Vassilvitskii (2007) explain K-means clustering performance is not gravely affected by the vast data volume. Due to the categorization nature of k-means clustering, unlike other algorithms in this thesis experiments, the data is not sub-sampled into several chunks. K-means can handle a lot of data and categorizes it into groups in which data points are similar to each other. This experiment aims to detect and distinguish groups of data points that contain a maximum occurrence of malicious and non-malicious data points. All training data is fed into the k-means clustering algorithm with varied k variable values to generate k models. All models are computed with hyper-parameters of 1000 random initializations, 15000 maximum number of iterations over the complete dataset and batch size of 256. The value of inertia is computed for each model k; the inertia is defined as the sum of squared distances (SSD) of samples to their nearest neighbor. This metric is used to select the best value of k hence producing the best model of k-means clustering. Figure 6.37 is an elbow-shaped plot of SSD against the number of clusters. According to the Figure 6.37 value of the k = 5 appears to be satisfactory.

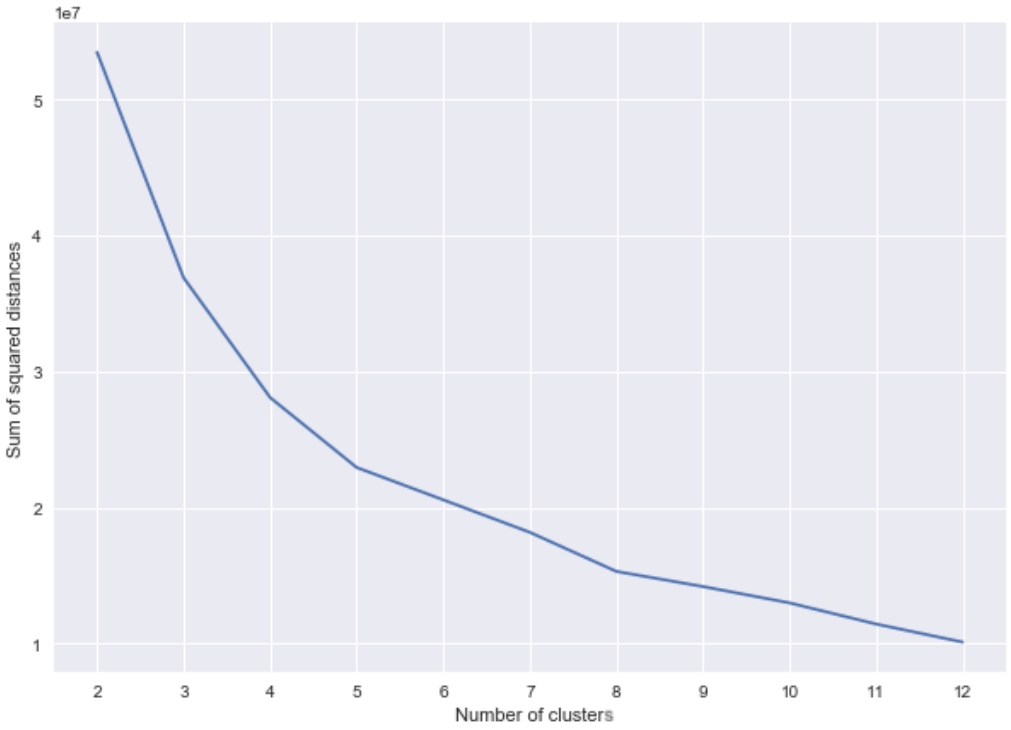

Figure 6.37. *Plot between k-cluster and sum of squared distances of samples to their closest cluster center.*

	Training Data	Test Data
Balanced Accuracy	0.8167213	0.8557088
Macro Precision	0.5000092	0.5000131
Macro Recall	0.8167213	0.8557088
Macro F1	0.4638692	0.4639096

Table 6.9. *Training and test data evaluation metrics for the model with K=5.*

The model with value k = 5 is evaluated with training data and test data. Table 6.9 describes the evaluation metrics statistics, and the k-means algorithm performed a decent job in separating the malicious data points from the non-malicious data points. Based on data of table 6.10 and table 6.11, it can be deduced that the model categorized the majority of malicious data-points in cluster 2 and cluster 4. On the contrary, the non-malicious data points are majorly designated to cluster 1, cluster 3, and cluster 5.

Cluster	Number of Malicious Data Points	Percentage of Malicious Data	Non-malicious Data Points	Percentage of Non-malicious Data
1	10	12.20%	6081677	25.13%
2	49	59.76%	2374469	9.81%
3	1	1.22%	9429675	38.97%
4	14	17.07%	888690	3.67%
5	8	9.76%	5423914	22.41%

Table 6.10. *Descriptive analysis of training data clustering for model with K=5.*

Cluster	Number of Malicious Data Points	Percentage of Malicious Data	Non-malicious Data Points	Percentage of Non-malicious Data
1	0	0.00%	1520828	25.14%
2	15	57.69%	592487	9.79%
3	1	3.85%	2358009	38.98%
4	7	26.92%	222614	3.68%
5	3	11.54%	1355663	22.41%

Table 6.11. *Descriptive analysis of test data clustering for model with K=5.*

As observed from the tables above, more than 55% of the anomalous data were clustered into one cluster along with a considerable amount of the remaining malicious data in another cluster. Nevertheless, the majority of non-malicious data was distributed in three clusters, with approximately 25%, 38%, and 22% of data in clusters, respectively, which sums up nearly 85% of the data.

A visual illustration of the clusters is created by performing principal component analysis (PCA) as a dimensionality reduction technique on the data. The data is transformed into three dimensions and then plotted into 2-dimensional and 3-dimensional space. The plot in figure 6.38 represents a 2-dimensional plot of test data with the first principal component as its x-axis and the second principal component as its y-axis. A different color represents all clusters in the plot with a black square representing the center of that cluster along with the number of that particular cluster. Red data points in the visual illustration depict the malicious data

points, and as seen in the plot, the majority of them lay in cluster 2 and cluster 4.

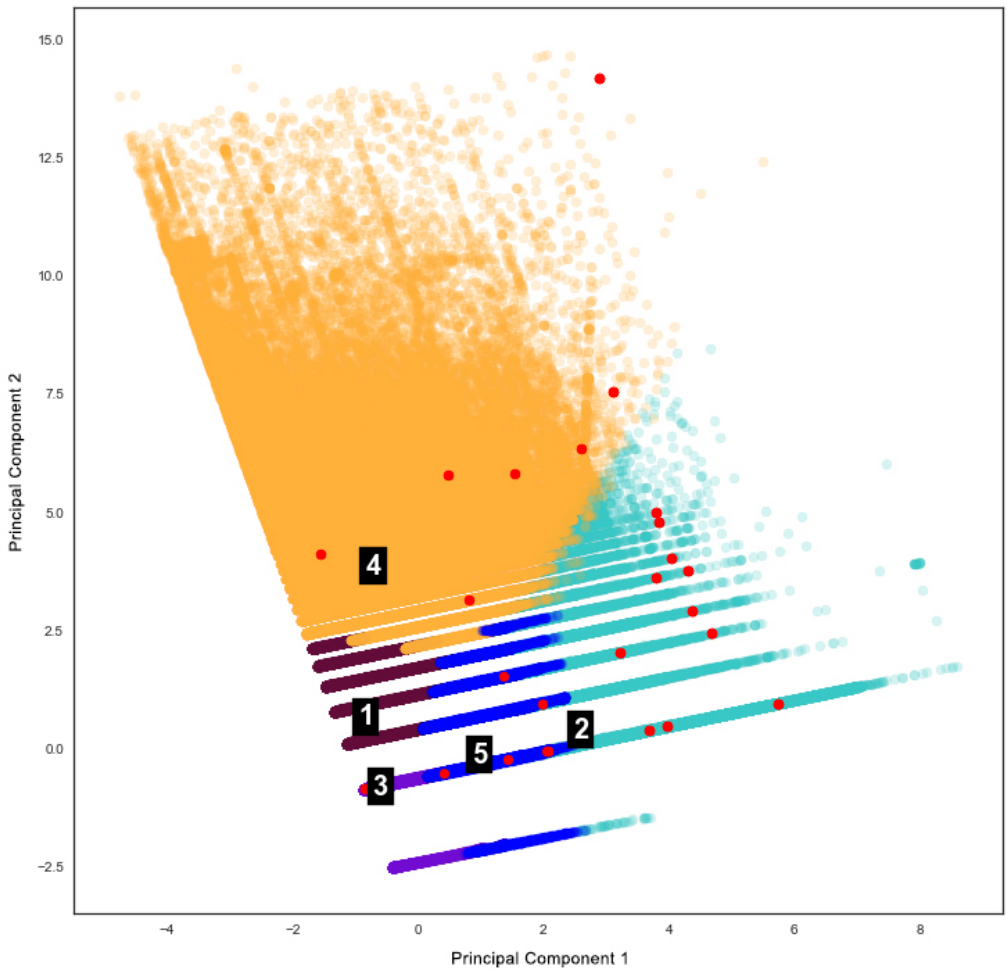

Figure 6.38. *2-dimensional plot of predicted test data representing all clusters data and their cluster centers.*

Figure 6.39 and Figure 6.40 are 3-dimensional space illustrations that give a clear understanding of how malicious and non-malicious data are clustered and distinguished from each other. In the 3-dimensional space, each axis is represented by a principal component extracted from the actual data.

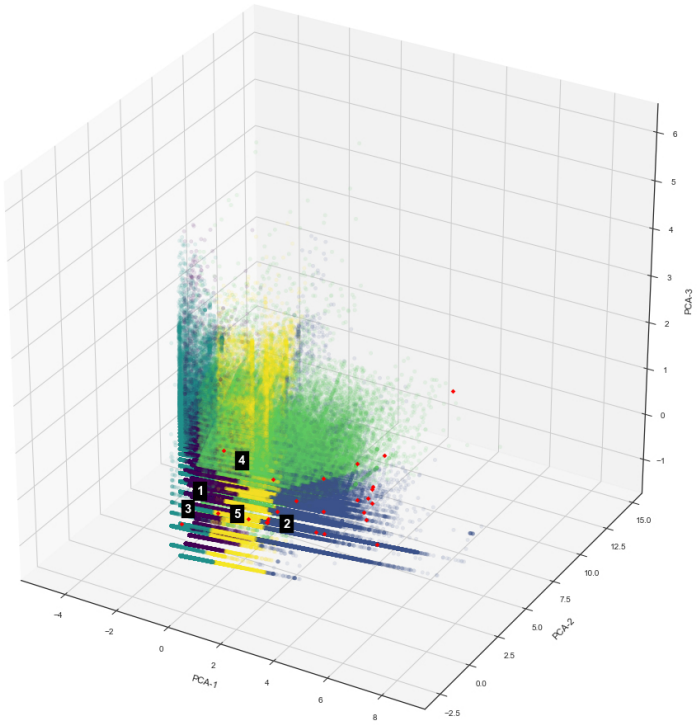

Figure 6.39. *3-dimensional plot of predicted test data representing all clusters data and their cluster centers (View 1).*

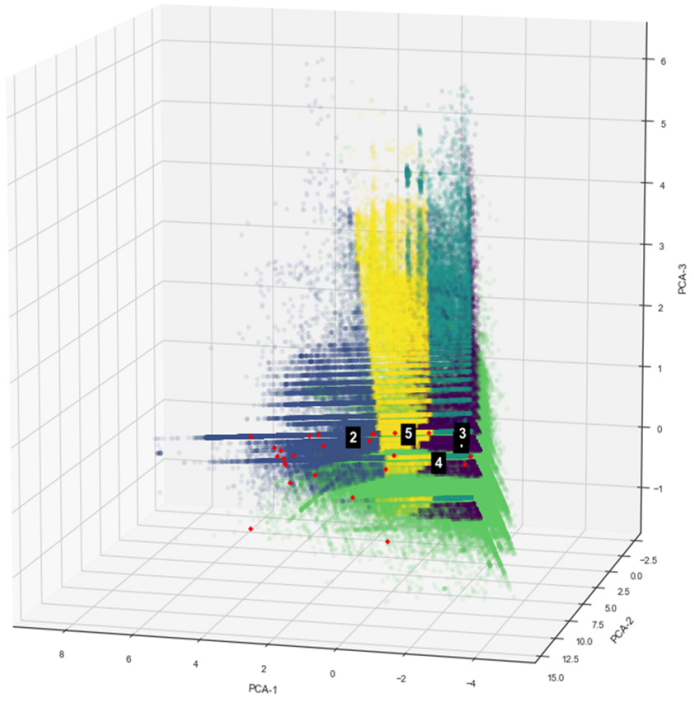

Figure 6.40. *3-dimensional plot of predicted test data representing all clusters data and their cluster centers (View 2).*

In order to create a conventional prediction confusion matrix, the clusters with the majority of malicious data points (Cluster 2 and Cluster 4) were merged into one. Conversely, clusters with the majority of non-malicious data points (Cluster 1, Cluster 3, and Cluster 5) were also merged into one. These merged clusters were labeled as malicious and non-malicious data points to compute a final confusion matrix. Based on the results, it can be observed that the k-means clustering model performed decently. The model with k = 5 was able to detect 77% (63 out of 82) of malicious data and 87% (20,935,266 out of 24,198,425) of non-malicious data in training data. It also successfully detected 85% (22 out of 26) of malicious data in test data along with 87% (5,234,500 out of 6,049,601) of non-malicious data, whereas the false-positives were only 15% (4 out of 22).

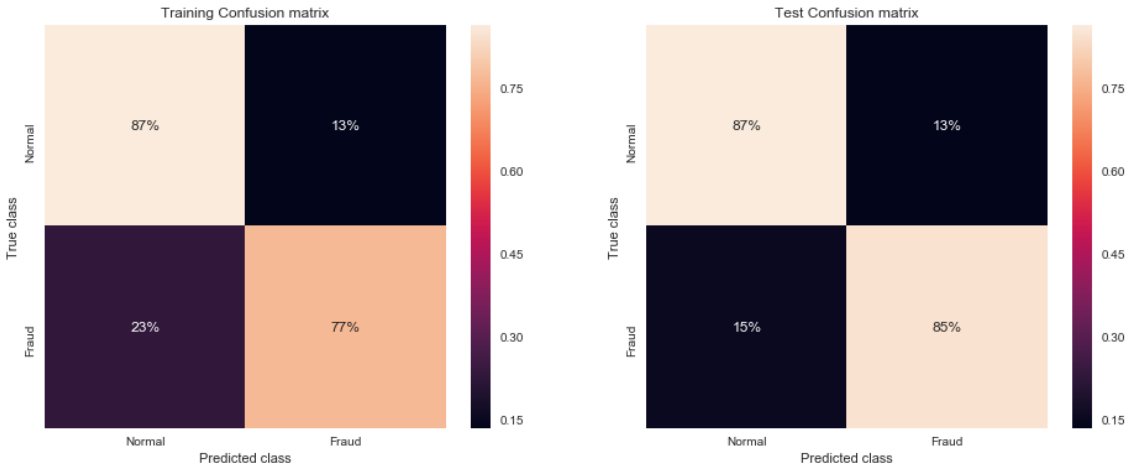

Figure 6.41. *Training and validation percentage confusion matrix.*

6.6 Deep Autoencoders

Canziani, Paszke, and Culurciello (2016) describe that Deep neural networks are capable of handling a large amount of data; however, the model computation time can be enormous. Pre-processed data is split into training and test sets and feed into the deep autoencoder network. In order to be able to find a hidden pattern of malicious data points within the imbalanced data set, the training set is over-sampled with 0.8% of synthetically generated malicious data points using Synthetic Minority Over-sampling Technique (SMOTE). Table 6.12 represents the structure of the neural network designed to encode and decode the data while trying to learn reconstruction.

Layer	Layer Type	Layer elements	Activation Function
Input	Input	7 attributes	(none)
Encoder	Dense	7 neurons	tanh
Encoder	Dense	7 neurons	tanh
Encoder	Dense	6 neurons	tanh
Encoder	Dense	4 neurons	tanh
Decoder	Dense	4 neurons	tanh
Decoder	Dense	6 neurons	tanh
Decoder	Dense	7 neurons	sigmoid
Output	Output	7 attributes	(none)

Table 6.12. *The network structure of deep autoencoder designed to detect anomalies.*

The autoencoder is trained with 100 epochs and a batch size of 256. The loss function selected for this use-case is mean squared logarithmic error (MSLE), as it only cares about the relative difference between the true and the predicted value, or in other words, it only cares about the percentual difference between them. Kingma and Ba (2014), *Adam* is used as an optimization algorithm with *tanh* and *sigmoid* as the activation functions. The choice of *tanh* for the encoder and decoder hidden layers is reasonable as it ranges between [-1,1] whereas, for the output layer, it is suitable to have data in a range of [0,1] where values between 0 and 1 represent the estimation of a data point being malicious. Table 6.13 describes the evaluation metrics obtained from the deep autoencoder model. Both training and test data were evaluated over the computed model to calculate these metrics.

	Training Data	Test Data
Balanced Accuracy	0.733119	0.803667
Macro Precision	0.716962	0.500029
Macro Recall	0.733119	0.803667
Macro F1	0.724655	0.488155
ROC AUC	0.870147	0.907563
Time	25075.109	177.6881

Table 6.13. *Evaluation results of training data and test data on a deep autoencoder network.*

Based on the table 6.13, the model is trained with 80% accuracy over the test data. The area under the curve for the receiver operating characteristic curve (ROC) is 90% for test data is illustrated in figure 6.42. The evaluation of the metrics table and ROC curve visualization gives a summary of how the model performs in general. From Figure 6.42, it can be deduced that the difference between the receiver operating characteristic curve (ROC) of training data and test data is not so significant. However, a higher AUC score for a model usually represents a better model. Recall of the autoencoder model shows a satisfying result as recall refers to the percentage of total relevant results correctly classified by the algorithm. In this use-case of detecting malicious and non-malicious data points, a higher recall metric is vital to the evaluation.

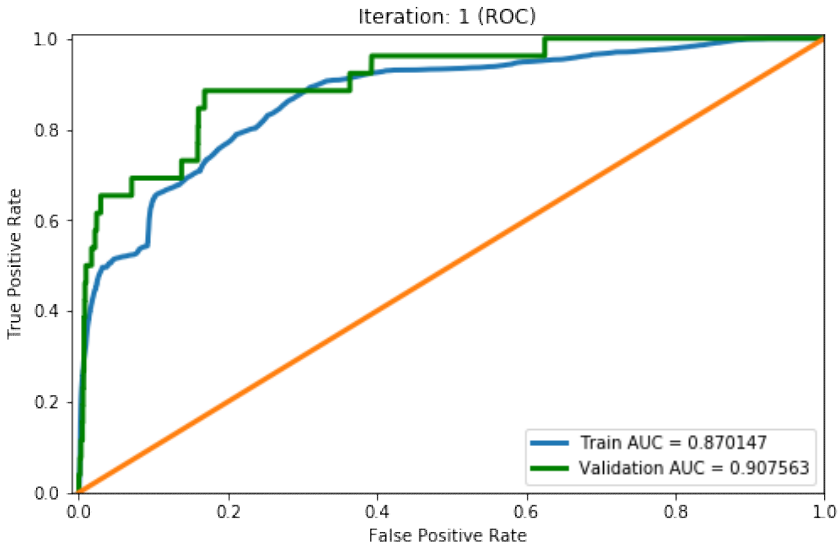

Figure 6.42. *Training and validation area under the curve (AUC) of receiver operating characteristic curve (ROC) for autoencoder model.*

Figure 6.43 represents the trade-off between precision and recall of the model at different thresholds for training and test data. Extracting a threshold is entirely based on the goal of our use-case. A value of threshold slightly below 4 seems reasonably satisfactory. Figure 6.44 is a visual illustration of reconstruction for model prediction based on predicted reconstruction errors. A threshold value slightly below 4 seems to cater to a significant number of malicious data points while not misclassifying majority of non-malicious data points.

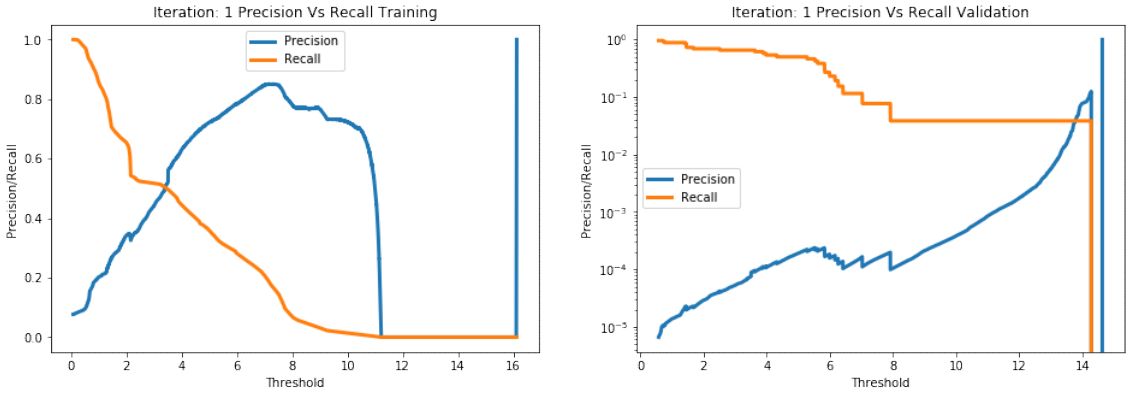

Figure 6.43. *Training and validation precision versus recall trade-off for autoencoder model.*

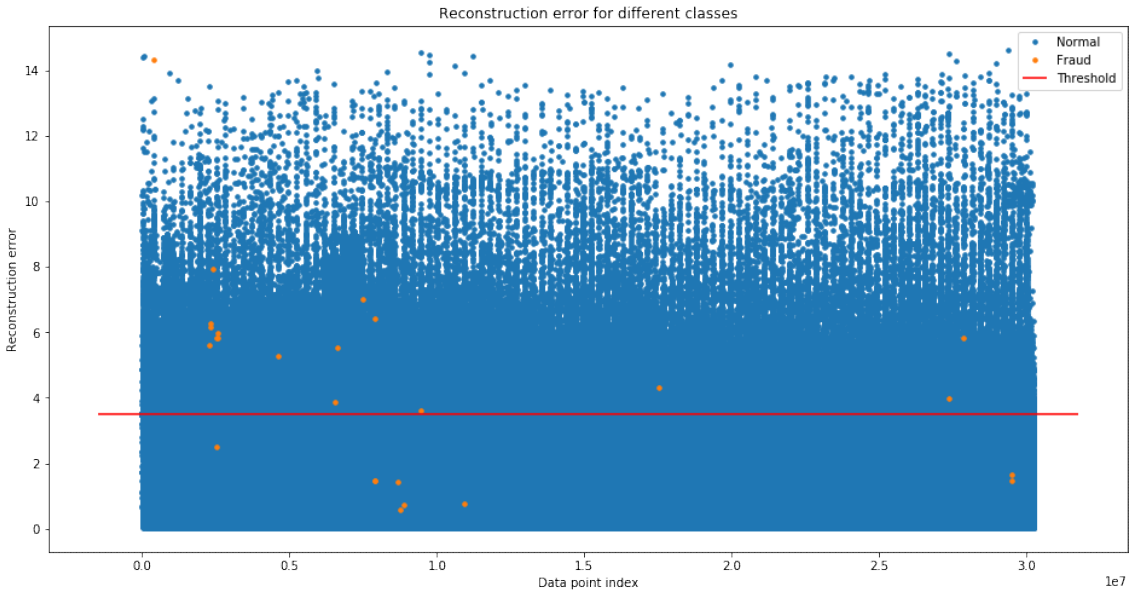

Figure 6.44. *Visual representation of reconstruction error of the test set from the model predicition to predict the malicious transactions.*

Based on the confusion matrix in Figure 6.45, it is apparent that autoencoder algorithm performed robustly. The computed model was able to successfully detect 65% (17 out of 26) of malicious data points in test data and only 50% (981,142 out of 1,976,769) of malicious data points in training data. The false-positive rate is higher in comparison to other experiments of this thesis research. The model misclassified approximately 50% (995,627 out of 1,976,769) of the data points as false-positive cases for training data and 35% (9 out of 26) for test data. However, the true-positives had a higher rate of 96% (Training: 23,294,998 out of 24,198,425 and Test: 5,823,535 out of 6,049,601) in both training and test data evaluation.

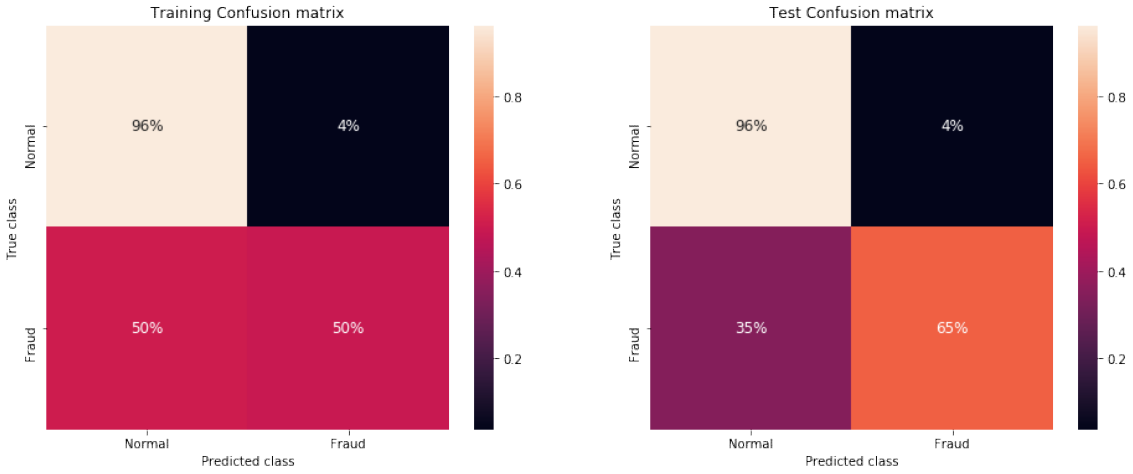

Figure 6.45. *Training and validation percentage confusion matrix for autoencoder model.*

6.7 Ensemble Classification

The test set used in all experiments is fed to an ensemble classifier. This ensemble classifier is composed by combining all the models computed for experiments in this thesis. The pre-processed test data consists of 26 malicious data points and 6,049,601 non-malicious data points. All the pre-computed model's from algorithms including Isolation Forest, Histogram based outlier score (HBOS), Cluster-based local outlier factor (CBLOF), Principal component analysis (PCA), K-means and Deep autoencoder are used to formulate a voting algorithm. Classification result from each algorithm is equally weighted, and as this use-case only requires an ensemble method for prediction, it is time-efficient.

	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	ROC AUC
Test Data	0.801402	0.500027	0.801402	0.48696	0.915766

Table 6.14. *Evaluation metrics for ensemble classification.*

Statistics computed after evaluating the ensemble classifier are shown in table 6.14. The ensemble classification method provides a robust result based on a majority voting algorithm. The area under the curve (AUC) of receiver operating characteristic curve (ROC) in figure 6.46 along with other statistics such as Macro-Recall as whole, states that the ensemble method performs significantly stable than previously evaluated algorithms.

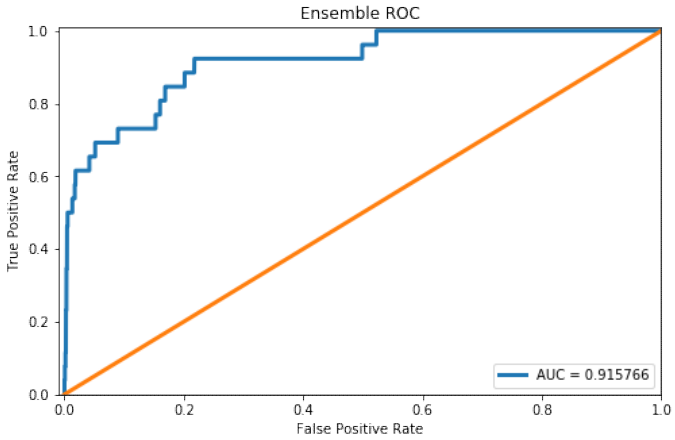

Figure 6.46. *Area under the curve (AUC) of receiver operating characteristic curve (ROC) for ensemble classification of all models.*

The evaluation was only performed for the test set as ensemble classification does not require its own training for this use-case. Figure 6.47 represents a visual illustration of a trade-off between precision and recall of the ensemble classification at

different thresholds. An optimal threshold is selected automatically based on an algorithm in the library we are utilizing for anomaly detection. Based on the confusion matrix in Figure 6.48 it is apparent that the ensemble classifier performed robustly and satisfactorily. 65% (17 out of 26) of the malicious data points were successfully detected, with only 35% (9 out of 26) of misclassification of false-positives. Equally important, 95% (5,740,820 out of 6,049,601) of non-malicious data points were correctly classified, with only 5% (308,781 out of 6,049,601) of true-negatives.

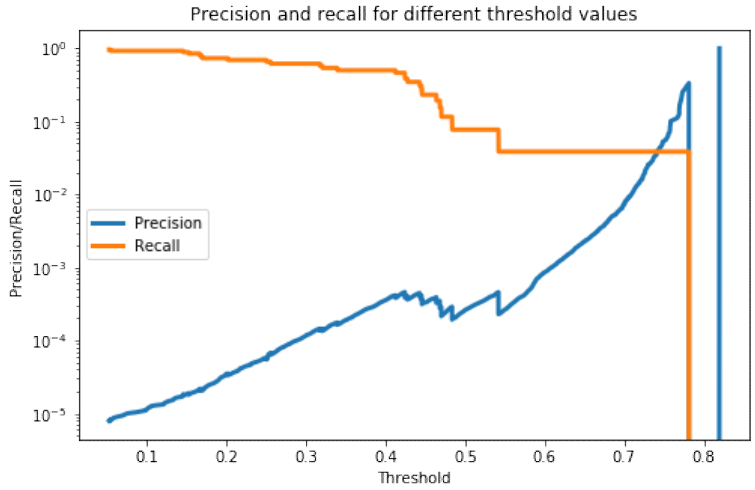

Figure 6.47. *Precision versus Recall trade-off for ensemble classification of all models.*

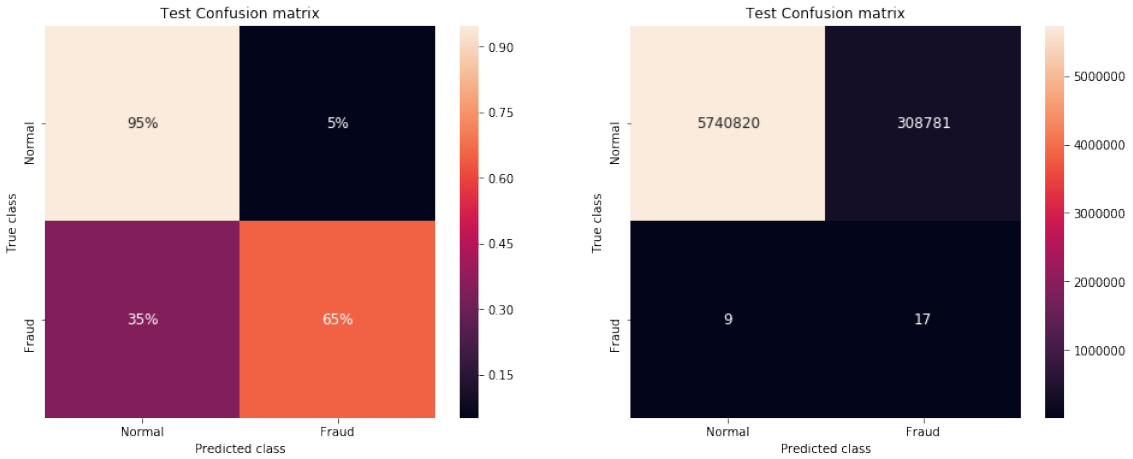

Figure 6.48. *Confusion matrix for ensemble classification of all models.*

7 Comparison

This chapter describes the comparison between the results of all experiments performed for this thesis. There are total of seven algorithms that are used to detect malicious and non-malicious transactions in the bitcoin data set used for this research. Table 7.1 describes the evaluation statistics computed for each algorithm. All evaluation metrics have their significance based on what kind of solution is required. This thesis research intends to maximize the malicious and non-malicious transaction detection while minimizing false-positive cases. Some algorithms perform better than the other, while ensemble classifier accounts for a vote from all algorithms to make a decision.

Classifer	Accuracy	Balanced-Accuracy	Macro-Precision	Macro-Recall	Macro-F1	ROC AUC	Precision	Recall	F1
IForest	0.969937	0.792662	0.500043	0.792662	0.492457	0.905661	0.000088	0.615385	0.000176
HBOS	0.871294	0.839493	0.500013	0.839493	0.465637	0.913317	0.000027	0.807692	0.000054
CBLOF	0.848902	0.789836	0.50001	0.789836	0.459159	0.818955	0.000021	0.730769	0.000042
PCA	0.881852	0.883234	0.500016	0.883234	0.468641	0.910491	0.000032	0.884615	0.000064
Kmeans	0.865264	0.855709	0.500013	0.855709	0.46391	NA	0.000027	0.846154	0.000054
Autoencoder	0.988407	0.744205	0.500092	0.744205	0.49727	0.884123	0.000185	0.5	0.000371
Ensemble	0.948957	0.801402	0.500027	0.801402	0.48696	0.915766	0.000055	0.653846	0.00011

Table 7.1. *Evaluation metrics of all algorithm models computed on test data.*

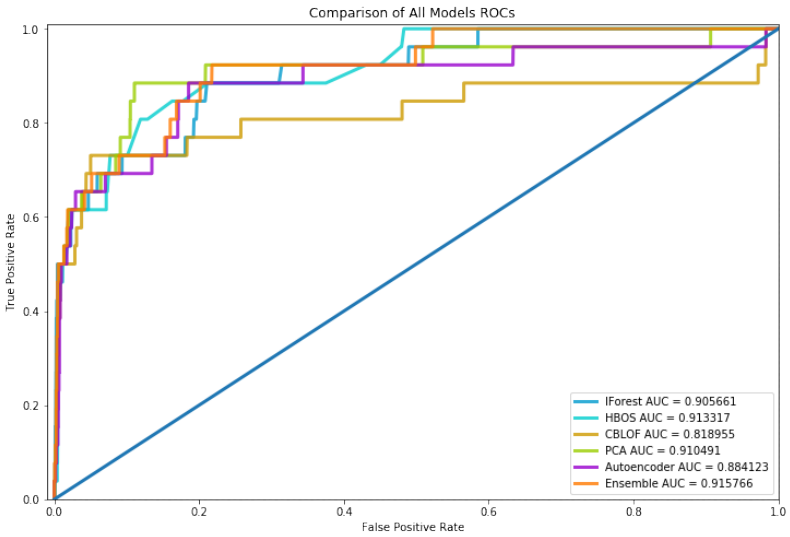

Figure 7.1. *Comparison of receiver operating characteristic (ROC) curve for all algorithms.*

Based on the tabular data from table 7.1 and a comparative illustration in figure 7.1 of the area under the curve of ROC along with the comparison of precision and recall plots in figure 7.2 and figure 7.3, it is difficult to determine the best choice.

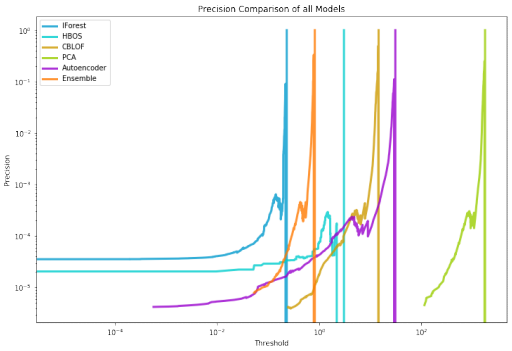

Figure 7.2. *Precision comparison.*

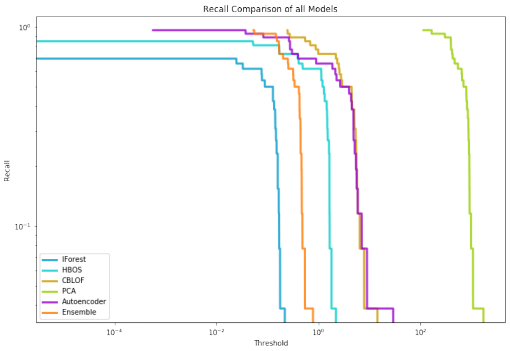

Figure 7.3. *Recall comparison.*

However, there is a trade-off between robustness and optimal performance. In terms of robustness ensemble method seems a better choice as it takes into account a deciding vote from six classifiers before making its decision, and this vote can have variable weights adjusted for each algorithm based on use-case. Nevertheless, for optimal solo performance, there is a tough competition between Isolation Forest, HBOS, K-means, Autoencoder, and PCA. Figure 7.4 is a histogram representing the accuracies of each model on the test set, solely relying only on accuracy for model selection is not wise in this particular use-case. Relying only on accuracy can be misleading.

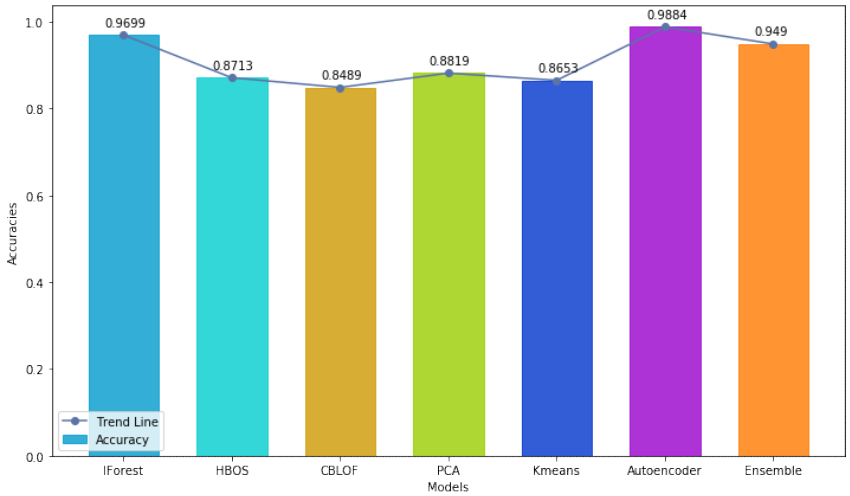

Figure 7.4. *Histogram comparing accuracies of all models.*

Macro evaluation metrics are useful indicators in scenarios where the overall performance of the system needs to be taken into consideration. A histogram represents a comparison of macro recall for all algorithms in Figure 7.5, and the histogram in Figure 7.6 illustrates the comparison of the macro-f1 scores for all algorithms. Based

on the visual illustrations of these macro evaluation metrics, ensemble method performs more stably and robustly. The robustness of an algorithm characterises how effective that algorithm performs on independent but similar data. However, isolation forest, k-means, principal component analysis (PCA), and autoencoder perform slightly better. The selection of one algorithm out of these is challenging and may require more in-depth use-case scenario information.

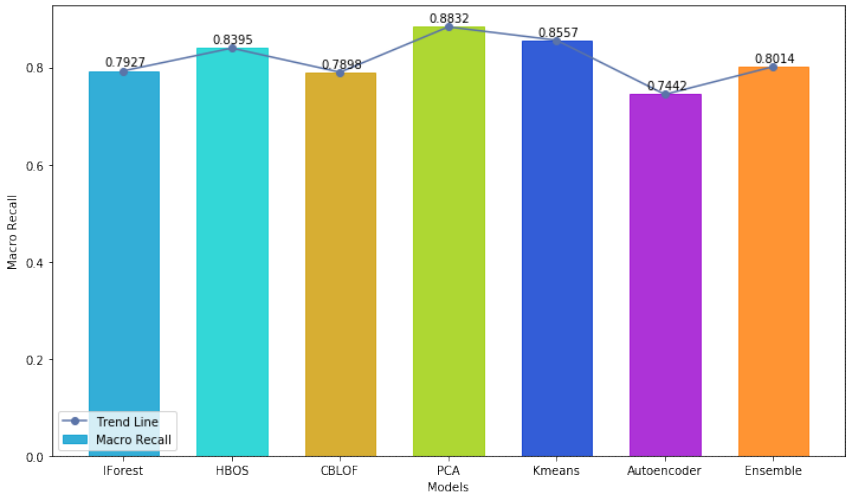

Figure 7.5. *Histogram comparing macro recall metric for all algorithms.*

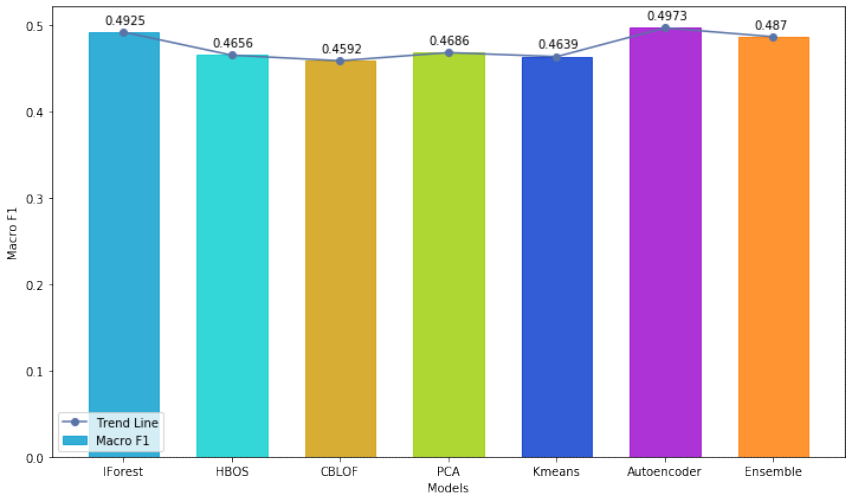

Figure 7.6. *Histogram comparing macro f1 metric for all algorithms.*

8 Conclusion

Evaluation of multiple unsupervised algorithms to detect anomalous behavior in an unspent transaction output (UTXO) based blockchain shows that malicious activities can be successfully identified. This goes in line with other research on anomaly detection in blockchains. If the evaluation metrics of previous researches were disclosed, the comparison would have been even more apparent. Although it can be indirectly compared to credit card fraud detection, the dynamics of data are entirely different and hence induce a variety of complex challenges.

It is possible that better results would have been obtained with more malicious data. The small size of malicious transactional data points in the dataset means that there is little anomalous patterns to recognize. An attempt was made to overcome this problem by synthetically generating malicious data points while making sure that it is not overdone. However, this affects the quality of anomaly detection, which leads to weaker performance of the models.

The selection of extracted features from the blockchain transaction graph to determine anomalous behavior seems an essential factor as certain variables may hold a unique prospect. Such as when treated as a time-series problem, it may disclose different solutions. However, this thesis research focuses on finding anomalous patterns using unsupervised learning techniques, and having a time-series element in the problem would increase the complexity of the scope.

Nonetheless, researchers have experimented with various unsupervised techniques to detect anomalous transactions in a blockchain and discovered somewhat successful ways to achieve it. However, the lack of relevant data and computing power limitations, along with algorithm's data handling capabilities, induce complex challenges.

Anomaly detection within blockchains can improve in the future if certain elements progress, such as the availability of relevant and rich data. Feature engineering and approaching the problem from another prospect such as time-series or network analysis can unlock new potential solutions. Additionally, distributed computing can be utilized to enhance algorithm's computing capabilities.

References

Arthur, David and Sergei Vassilvitskii (2007). "k-means++: The advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, pp. 1027–1035.

Baird, Leemon (2016). *The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance*. English.

Barrera, Alex (Apr. 2014). *A Guide to Bitcoin (Part 1 and 2): A look under the hood*. URL: http://tech.eu/features/808/bitcoin-part-one/ (visited on 10/16/2018).

Bayer, Dave, Stuart Haber, and W Scott Stornetta (1993). "Improving the efficiency and reliability of digital time-stamping". In: *Sequences II*. Springer, pp. 329–334.

Bentley, Jon Louis (1975). "Multidimensional binary search trees used for associative searching". In: *Communications of the ACM* 18.9, pp. 509–517.

Bitcoin Forum (2014). *Bitcoin Forum*. https://bitcointalk.org/index.php?topic=576337. [Online; accessed 7-September-2019].

Bitcoin-Wiki (May 2016). *Proof of work*. URL: https://en.bitcoin.it/wiki/Proof_of_work (visited on 08/30/2018).

Blockchain.com (2018). *Stone Man Loss Transaction Visualization*. [Online; accessed 12-November-2018]. URL: https://www.blockchain.com/btc/tree/120749.

— (2019). *Blockchain.com*. https://www.blockchain.com/en/charts/n-transactions-total?timespan=all. [Online; accessed 7-September-2019].

Bogner, Andreas (2017). "Seeing is understanding: anomaly detection in blockchains with visualized features". In: *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*. ACM, pp. 5–8.

Bolton, Richard J, David J Hand, et al. (2001). "Unsupervised profiling methods for fraud detection". In: *Credit Scoring and Credit Control VII*, pp. 235–255.

Breiman, Leo (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32.

Breunig, Markus M et al. (2000). "LOF: identifying density-based local outliers". In: *ACM sigmod record*. Vol. 29. 2. ACM, pp. 93–104.

Brito, Jerry and Andrea Castillo (2013). "Bitcoin: A primer for policymakers". In: *Policy: A Journal of Public Policy and Ideas* 29.4, pp. 3–12.

Brugere, Ivan (2013). *Bitcoin Transaction Network Dataset*. [Online; accessed 29-November-2018]. URL: http://compbio.cs.uic.edu/data/bitcoin/.

Burkhardt, Daniel, Maximilian Werling, and Heiner Lasi (2018). "Distributed Ledger". English. In: IEEE, pp. 1–9.

Buterin, Vitalik (Aug. 6, 2016). *On Public and Private Blockchains*. URL: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/ (visited on 09/26/2018).

Calvez, Antoine Le (2015). *bitcoin-blockchain-parser*. https://github.com/alecalve/python-bitcoin-blockchain-parser.

Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello (2016). "An analysis of deep neural network models for practical applications". In: *arXiv preprint arXiv:1605.07678*.

Carlozo, Lou (2017). "What is blockchain?" In: *Journal of Accountancy* 224.1, p. 29.

Chandola, Varun, Arindam Banerjee, and Vipin Kumar (2009). "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3, p. 15.

Chawla, Nitesh V et al. (2002). "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16, pp. 321–357.

Chen, Lei, M Tamer Özsu, and Vincent Oria (2005). "Robust and fast similarity search for moving object trajectories". In: *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, pp. 491–502.

Christin, Nicolas (2013). "Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace". In: *Proceedings of the 22nd international conference on World Wide Web*. ACM, pp. 213–224.

Commons, Wikimedia (2016). *File:Autoencoder structure.png — Wikimedia Commons, the free media repository*. [Online; accessed 1-September-2019]. URL: https://commons.wikimedia.org/w/index.php?title=File:Autoencoder_structure.png&oldid=185174476.

Cuesta-Albertos, Juan Antonio, Alfonso Gordaliza, Carlos Matrán, et al. (1997). "Trimmed k-means: An attempt to robustify quantizers". In: *The Annals of Statistics* 25.2, pp. 553–576.

Delgado-Segura, Sergi et al. (2018). "Analysis of the Bitcoin UTXO set". In: *Proceedings of the 5th Workshop on Bitcoin and Blockchain Research Research (in Assocation with Financial Crypto 18), Lecture Notes in Computer Science*.

European-Central-Bank (2014). *European Central Bank. Report on card fraud*. URL: https://www.ecb.europa.eu/press/pr/date/2014/html/pr140225.en.html (visited on 11/03/2018).

Farren, Derek, Thai Pham, and Marco Alban-Hidalgo (2016). "Low Latency Anomaly Detection and Bayesian Network Prediction of Anomaly Likelihood". In: *arXiv preprint arXiv:1611.03898*.

Friedman, Jerome, Trevor Hastie, Robert Tibshirani, et al. (2000). "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)". In: *The annals of statistics* 28.2, pp. 337–407.

Goldstein, Markus and Andreas Dengel (2012). "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm". In: *KI-2012: Poster and Demo Track*, pp. 59–63.

Goldstein, Markus and Seiichi Uchida (2016). "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data". In: *PloS one* 11.4, e0152173.

Haber, Stuart and W Scott Stornetta (1990). "How to time-stamp a digital document". In: *Conference on the Theory and Application of Cryptography*. Springer, pp. 437–455.

He, Zengyou, Xiaofei Xu, and Shengchun Deng (2003). "Discovering cluster-based local outliers". In: *Pattern Recognition Letters* 24.9-10, pp. 1641–1650.

Hecht-Nielsen, Robert (1992). "Theory of the backpropagation neural network". In: *Neural networks for perception*. Elsevier, pp. 65–93.

Heilman, Ethan et al. (2015). "Eclipse Attacks on Bitcoin's Peer-to-Peer Network." In: *USENIX Security Symposium*, pp. 129–144.

Henderson, Keith et al. (2012). "Rolx: structural role extraction & mining in large graphs". In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 1231–1239.

Hirshman, Jason, Yifei Huang, and Stephen Macke (2013). *Unsupervised approaches to detecting anomalous behavior in the bitcoin transaction network*. Tech. rep. Technical report, Stanford University.

Huang, Butian et al. (2017). "Behavior pattern clustering in blockchain networks". In: *Multimedia Tools and Applications* 76.19, pp. 20099–20110.

Iansiti, Marco and Karim R. Lakhani (2017). *The Truth About Blockchain*. English.

Ibanez, Luis-Daniel et al. (2017). "Redecentralizing the Web with Distributed Ledgers". English. In: *IEEE Intelligent Systems* 32.1, pp. 92–95.

Jolliffe, Ian (2011). *Principal component analysis*. Springer.

Kim, Yoohwan et al. (2004). "PacketScore: Statistics-based overload control against distributed denial-of-service attacks". In: *IEEE INFOCOM 2004*. Vol. 4. IEEE, pp. 2594–2604.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Krebs, Brian (2013). *Adobe hacked: customer data, source code compromised*. URL: https://www.smh.com.au/technology/adobe-hacked-customer-data-source-code-compromised-20131004-hv1wl.html (visited on 11/05/2018).

Lamport, Leslie, Robert Shostak, and Marshall Pease (July 1982). "The Byzantine Generals Problem". In: *ACM Transactions on Programming Languages and Systems* 4/3, pp. 382–401. URL: https://www.microsoft.com/en-us/research/publication/byzantine-generals-problem/.

Lehmann, Erich L and George Casella (2006). *Theory of point estimation*. Springer Science & Business Media.

Leskovec, Jure, Jon Kleinberg, and Christos Faloutsos (2007). "Graph evolution: Densification and shrinking diameters". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1, p. 2.

Lin, Iuon-Chang, Tzu-Chun Liao, and Iuon-Chang Lin (2017). "A Survey of Blockchain Security Issues and Challenges". English. In: *International Journal of Network Security* 19.5, pp. 653–659.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou (2008). "Isolation forest". In: *2008 Eighth IEEE International Conference on Data Mining*. IEEE, pp. 413–422.

MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, pp. 281–297. URL: https://projecteuclid.org/euclid.bsmsp/1200512992.

MacQueen, James et al. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA, pp. 281–297.

McCurry, Justin (2016). *100 thieves steal USD 13m in three hours from cash machines across Japan*. URL: https://www.theguardian.com/world/2016/may/23/japan-cash-machine-100-thieves-steal-13m-dollars-three-hours (visited on 11/05/2018).

Milutinovi, Monia (2018). "Cryptocurrency". English. In: *Ekonomika* 64.1, pp. 105–122.

Monamo, Patrick M, Vukosi Marivate, and Bhesipho Twala (2016). "A Multifaceted Approach to Bitcoin Fraud Detection: Global and Local Outliers". In: *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*. IEEE, pp. 188–194.

Monamo, Patrick, Vukosi Marivate, and Bheki Twala (2016). "Unsupervised learning for robust Bitcoin fraud detection". In: *Information Security for South Africa (ISSA), 2016*. IEEE, pp. 129–134.

Morse, Michael D and Jignesh M Patel (2007). "An efficient and accurate method for evaluating time series similarity". In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, pp. 569–580.

Nakamoto, Satoshi (2008). "Bitcoin: A peer-to-peer electronic cash system". In:

North-Denver-News (2015). *Cybercrime- what are the costs to victims*. URL: http://northdenvernews.com/cybercrime-costs-victims (visited on 11/03/2018).

Othman, Zulaiha Ali et al. (2014). "Clustering Network Intrusion Detection". In: *Journal of Applied Sciences* 14.10, pp. 969–980.

Patil, Valmik Ravindra et al. (2018). "Bitcoin Fraud Detection using Data Mining Approach". In: *Journal of Information Technology and Sciences* 4.2.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pham, Thai and Steven Lee (2016a). "Anomaly detection in bitcoin network using unsupervised learning methods". In: *arXiv preprint arXiv:1611.03941*.

— (2016b). "Anomaly Detection in the Bitcoin System-A Network Perspective". In: *arXiv preprint arXiv:1611.03942*.

Phua, Clifton et al. (2010). "A comprehensive survey of data mining-based fraud detection research". In: *arXiv preprint arXiv:1009.6119*.

Preiss, Bruno R (2008). *Data structures and algorithms with object-oriented design patterns in C++*. John Wiley & Sons.

Raschka, Sebastian (Apr. 2018). "MLxtend: Providing machine learning and data science utilities and extensions to Pythons scientific computing stack". In: *The Journal of Open Source Software* 3.24. DOI: 10.21105/joss.00638. URL: http://joss.theoj.org/papers/10.21105/joss.00638.

Raval and Siraj (2016). *What Is a Decentralized Application?* Decentralized Applications: Harnessing Bitcoin's Blockchain Technology. O'Reilly Media, Inc. ISBN: 978-1-4919-2452-5.

Reid, Fergal and Martin Harrigan (2011). "An analysis of anonymity in the bitcoin system". In: *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*. IEEE, pp. 1318–1326.

— (2013). "An analysis of anonymity in the bitcoin system". In: *Security and privacy in social networks*. Springer, pp. 197–223.

Sajana, P, M Sindhu, and M Sethumadhavan (2018). "On Blockchain Applications: Hyperledger Fabric And Ethereum". In: *International Journal of Pure and Applied Mathematics* 118.18.

Sandle, P and P Char (2014). "Cyber crime costs global economy 445 billion USD a year: Report". In: *Reuters*.

Sarkar, Saroje K, Habshah Midi, and Sohel Rana (2011). "Detection of outliers and influential observations in binary logistic regression: An empirical study". In: *Journal of Applied Sciences* 11.1, pp. 26–35.

Shyu, Mei-Ling et al. (2003). *A novel anomaly detection scheme based on principal component classifier*. Tech. rep. MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL and COMPUTER ENGINEERING.

Signorini, Matteo et al. (2018). "BAD: Blockchain Anomaly Detection". In: *arXiv preprint arXiv:1807.03833*.

Skvorc, Bruno (2018). *Proof of Stake vs Proof of Work*. URL: https://www.sitepoint.com/proof-of-stake-vs-proof-of-work/ (visited on 09/30/2018).

Smith, S (2015). *Cybercrime will Cost Businesses over 2 Trillion USD by 2019*.

Stone-Man (2010). *Stone Man Loss*. URL: https://bitcointalk.org/index.php?topic=782.5 (visited on 11/11/2018).

The-Economist (2015). *The great chain of being sure about things*. URL: https://www.economist.com/briefing/2015/10/31/the-great-chain-of-being-sure-about-things (visited on 10/01/2018).

Valenta, Martin and Philipp Sandner (2017). *Comparison of Ethereum, Hyperledger Fabric and Corda*. URL: http://explore-ip.com/2017_Comparison-of-Ethereum-Hyperledger-Corda.pdf (visited on 09/30/2018).

Veronese, Giuliana S. et al. (2013). "Efficient Byzantine Fault-Tolerance". English. In: *IEEE Transactions on Computers* 62.1, p. 16.

Vlachos, Michail, George Kollios, and Dimitrios Gunopulos (2002). "Discovering similar multidimensional trajectories". In: *Data Engineering, 2002. Proceedings. 18th International Conference on*. IEEE, pp. 673–684.

Wang, Yasi, Hongxun Yao, and Sicheng Zhao (2016). "Auto-encoder based dimensionality reduction". In: *Neurocomputing* 184, pp. 232–242.

Wikimedia-Commons (2018). *File:Bitcoin October 2013.png — Wikimedia Commons, the free media repository*. [Online; accessed 11-November-2018]. URL: https://commons.wikimedia.org/w/index.php?title=File:Bitcoin_October_2013.png&oldid=320518959.

Wikipedia (July 2018). *Gossip protocol*. URL: https://en.wikipedia.org/wiki/Gossip_protocol (visited on 09/10/2018).

Zambre, Deepak and Ajey Shah (2013). "Analysis of bitcoin network dataset for fraud". In: *Unpublished Report*.

Zheng, Jin and Lihui Peng (2018). "An autoencoder-based image reconstruction for electrical capacitance tomography". In: *IEEE Sensors Journal* 18.13, pp. 5464–5474.

Zheng, Zibin et al. (2017). "Blockchain Challenges and Opportunities: A Survey". Work Pap.

Zimek, Arthur and Erich Schubert (2017). "Outlier detection". In: *Encyclopedia of Database Systems*, pp. 1–5.